المدخل الى لغة الفورتران



د. سالم أحمد سحّاب

د. محمد يحيى عبد الرحمن

الناشر

مركز النشر العلمي

جامعة الملك عبد العزيز

جدة ١٤٠٦هـ (١٩٨٦م)

المدخل إلى لغة الفورتران

تأليف

د. سالم أحمد سحاب

أستاذ مساعد - قسم الرياضيات
كلية العلوم - جامعة الملك عبد العزيز

د. محمد يحيى عبد الرحمن

أستاذ مساعد - قسم علوم الحاسبات
كلية العلوم - جامعة الملك عبد العزيز

الناشر
مركز النشر العلمي جامعة الملك عبد العزيز
ص.ب ١٥٤٠، جدة ٢١٤٤١
المملكة العربية السعودية

بسم الله الرحمن الرحيم

بسم الله الرحمن الرحيم

يَرْفَعُ اللهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَات .

" صدق الله العظيم "

© ١٤٠٦هـ (١٩٨٦م) جامعة الملك عبد العزيز .

جميع حقوق طبع هذا البحث محفوظة وملك الجامعة . مسموح بخزنه في أي بنك للمعلومات والاقتباس منه دون إذن من صاحب الحق . غير مسموح بطبعه كاملاً ، أو نقله على أية هيئة أو بأية وسيلة ، سواء كانت إلكترونية ، أو شرائط ممغنطة ، أو ميكانيكية ، أو استنساخاً ، أو تسجيلاً ، أو غير ذلك من الوسائل إلا بإذن كتابي من صاحب حق الطبع .

الطبعة الأولى ١٤٠٦هـ (١٩٨٦م)

مطابع جامعة الملك عبدالعزيز

إهداء

إلى والدينا :

عليهما من الله سحائب الرحمة

وأمطار الغفران .

المؤلفان

بسم الله الرحمن الرحيم

الحمد لله والصلاة والسلام على سيدنا محمد وعلى آله وصحبه ومن سار على هديه الى يوم الدين ، وبعد :

فإن الحاجة ماسة جداً لتأليف كتاب باللغة العربية يغطي احتياجات معظم طلاب الجامعات العربية وطلاب الكليات العلمية خاصة في مجال دراسة البرمجة باستعمال الفورتران «FORTRAN» والتي تعد من أهم اللغات المتداولة في مجال البرمجة على الحاسب الآلي بل أولها وأشملها على الاطلاق من حيث وفرتها العلمية وإمكانياتها التطبيقية في شتى مجالات العلوم .

ومن هذا المنطلق عزمنا مستعينين بالله متكلين عليه على تأليف هذا الكتاب وتقديمه كأحد الكتب الممكن أن تؤدي للطالب أن شاء الله الغرض المنشود من دراسة مبدئية في علم الحاسبات وتزوده بالمعلومات الأساسية والضرورية لتغطية منهج عن لغة الفورتوان وبعض تطبيقاتها العلمية في مدة زمنية تعادل فصلاً دارسياً واحداً . كما أن الكتاب سيساعد على إعطاء الطالب المتخصص القاعدة العريضة الصلبة التي يحتاجها في مجال التخصص في علوم الحاسبات الآلية إن شاء الله .

وقد روعي في هذا الكتاب أن لايكون له متطلب سابق على مستوى الجامعة وبمعنى آخر فإن كل ما يحتاجه الطالب أو الدارس لهذا الكتاب هو الإلمام بمبادىء علم الجبر أو مبادىء الرياضيات الحديثة التي تدرس غالبا في المدارس الثانوية .

أما أسلوب الكتاب فهو يعتمد على محاولة التوفيق بين استيعاب الطالب للمادة العلمية النظرية وبين زيادة مقدرته على البرمجة الفعلية وتطبيق الناحية النظرية في أسلوب متدرج بسيط ينمي قدرات الطالب ويوجد الحافز المشجع الذي يدفع الطالب قدما لقضاء وقت أطول بين جنبات المعمل عن طواعية ورغبة في استيعاب المزيد عن امكانيات اللغة وتطبيقاتها الواسعة فضلاً عن الرغبة في انجاز مايوكل اليه من كتابة وبرمجة بعض المشكلات والمسائل المطلوبة منه .

وفي الختام نسأل الله العلي القدير أن يجعل هذا العمل المتواضع البسيط لبنة بناءة من لبنات المكتبة العربية العلمية التي تساير تطور الأمتين العربية والاسلامية في مجالات العلوم الحديثة والله من وراء القصد .

المؤلفان

محتويات الكتاب

رقم الصفحة

الفصل الثامن : البرامج الجزئية .

الفصـل الأول
تطور الحاسـبات والبرمجـة

الفصـل الأول

١-١ تطور الحاسبات الألكترونية العددية Electronic Digital Computers

لن ندهش كثيرا اذا علمنا أن عصر الحاسبات قد بدأ التفكير فيه في سنوات ماقبل الميلاد (من سنة ٤٠٠٠ الى سنة ٣٠٠٠ قبل الميلاد) ، حيث بدأ التفكير فيها باستخدام مستطيل معدني مركب عليه خيوط أو أسلاك متوازية وفي كل منها عدد من الكرات الصغيرة ذات الألوان المختلفة حيث كل لون منها يمثل قوة حسابية معينة (آحاد – عشرات ... الخ) ، وقد اطلق على هذه الالآت اسم الحاسبات calculi .

وفي منتصف القرن السادس عشر امكن تصميم أولي الحاسبات الميكانيكية والتي تعتمد في إجراء عمليات الجمع والطرح على مجموعة من التروس والبكرات ، وأمكن تحسينها بعد ذلك بحيث يمكنها اجراء عمليات الضرب بطريقة مباشرة بعد أن كانت تتم بأجراء عمليات جمع متتالية .

وفي حوالي سنة ١٨٣٠ أمكن للرياضي البريطاني شارل بباج Charles Babbage أن يضع تصوراً لآلة حاسبة تحتوي على :

وسيلة إدخال بيانات – امكانية تخزين – امكانيات عمل بعض الحسابات – وسيلة إخراج نتائـج . وقد اطلق على آلته اسم الآلة التحليلية The analytical engine .

وبعد حوالي مائة عام أمكن تحقيق تصورات ذلك العالم البريطاني عندما بدىء في تصميم وبناء الحاسب الذي سمى MARK I ، وكان ذلك في جامعة هارفارد عام ١٩٣٩م وفي عام ١٩٤٤م بدىء في تشغيل ذلك الحاسب وأعتبر هذا العام كفاتحة لعصر الحاسبات . ومع أن هذا الحاسب كان في البداية يعتمد في تصميمه على دوائر كهربائية ميكانيكية الَّا أنه أمكن بعد ذلك تطويره بادخال الصمامات في دوائرة الكهربية مما جعله أسرع وأطلق عليه حينذاك اسم ENIAC وكان ذلك أول حاسب الكتروني رقمي (Digital) . وكانت العقبة التي واجهت مثل تلك الحاسبات في ذاك الوقت هو تخزين التعليمات والبيانات اللازمة لكل مشكلة يراد حلها ، الى أن أمكن التغلب على تلك العقبة بعد سنوات قليلة بأستخدام نظام البطاقات المثقوبة والذي كان قد ابتكره الاحصائي هيرمان هيوليرث Herman Hollerith عام ١٨٨٩م – وقد اطلق على أول نوع من تلك الحاسبات التي تحوي دوائر تخزين بيانات ومعلومات بأسم UNIVACI ، وظلت الصمامات الكهربية هي المكون الأساسي للحاسبات حتى أواخر عام ١٩٥٠م حيث ظهرت المقومات والتي تعرف بأسم الترانزستور والتي كانت البديل الأمثل للصمامات سواء في جعل الحاسبات أقل حجماً وتكلفة وأقل انبعاثا

للحرارة التي كانت تحدثها الصمامات والتي بالتالي كانت تتطلب تكلفة أكثر في تجهيزاته . وقد اطلق على الفترة من ١٩٥٠م الى ١٩٦٥م بأسم فترة الجيل الثاني من الحاسبات .

وقد أمكن بعد ذلك تطوير هذا الجيل من الحاسبات بأدخال نظام الدوائر المتكاملة Integrated Circuits والذي ساعد كثيراً في تقليل حجم الحاسبات وتكلفتها وزيادة سرعة كفائتها في اجراء العمليات الحسابية كما أمكن ادخال تعديلات كبيرة في تصميماتها الداخلية والخارجية . وقد كانت الحاسبات من طراز أ. ب. م IBM 360,IBM 370 هي رائدة هذا الجيل من الحاسبات والذي أطلق عليه الجيل الثالث لتطور الحاسبات .

وفي وقتنا الحاضر والذي يعتبر الجيل الرابع للحاسبات ادخلت كثير من التعديلات في تصميمات الحاسبات وذلك بفضل التقدم الذي يحدث حالياً في مجال تصنيع الدوائر المتكاملة والذي أدى الى امكانية انقاص حجم الحاسبات وتقليل تكلفتها وزيادة حجم التعليمات والبيانات التي يمكن تخزينها وزيادة سرعة إجراء العمليات الحسابية والمنطقية التي يقوم بها .

ولا شك أننا نلمس الآن مدى ما أحدثته الحاسبات من تقدم في جميع المجالات سواء الحربية (العسكرية) أو المدنية حتى أصبحت العصب الرئيسي لكثير من الأنشطة والقطاعات لأية دولة متقدمة ، خذ مثلاً الأسلحة العسكرية المتطورة فالصواريخ والغواصات والطائرات الحربية والأقمار الصناعية وحتى الدبابات الحديثة وغيرها من الأسلحة تعتمد على الحاسب الألكتروني الى درجة عالية جداً بحيث أنها تفقد فعاليتها اذا أختل عمل الحاسب الألكتروني لسبب أو لآخر ، بل أن بعض طائرات التجسس الحديثة تعتمد إعتماداً كلياً على الحاسب الألكتروني في أداء مهمتها التصويرية والتجسسية .

أما من الناحية المدنية فلا يكاد يخلو قطاع واحد من قطاعات الدولة إلّا وهو ماض قدماً في سبيل إدخال الحاسب الألكتروني لتسهيل عمل ذلك القطاع ومن أهم هذه القطاعات وزارة الداخلية الذي من خططه الحالية تخزين المعلومات الأساسية عن جميع أفراد الشعب السعودي بواسطة الحاسب الألكتروني لتحل محل إستعمال الملفات التقليدية التي تستهلك الكثير من الجهد والوقت بالأضافة الى إمكانية تلفها أو فقدانها وإستيعابها لمساحات كبيرة لحفظها .

ومن الأمثلة الأخرى إعتماد شركات الطيران الحديثة ومنها السعودية على الحاسب الألكتروني الذي يقوم بأعطاء جميع المعلومات الضرورية عن الحجز وأرقام الرحلات ومواعيد الاقلاع والوصول ... الخ ، والأمثلة كثيرة وعديدة في المجالات العلمية المختلفة كالطب والهندسة وعلوم الفضاء .

١-٢ الوحدات الأساسية المكونة للحاسبات العددية :

لنفرض أننا نعيش الفترة التي سبقت ظهور الحاسبات وطلب منا حل أحدى المشكلات الرياضية العددية ، وليس لدينا من أدوات لحل تلك المشكلة سوى الورقة والقلم . فلكي نبدأ الحل لابد من توافر البيانات الخاصة بالمشكلة وبتطبيق تلك البيانات على طريقة الحل سيمكننا الحصول على النتائج المطلوبة .

فعلى سبيل المثال ، لنفرض أننا نريد حساب Y والتي تعطي من العلاقتين .

$$X = A + B - C, Y = \frac{X}{D}$$

فلحساب قيمة Y فأن ذلك يتطلب عدة تساؤلات أساسية منها :-

١) ماهي المتغيرات التي لابد من معرفة قيمتها كي تؤدي في النهاية الى معرفة قيمة Y ؟ .
٢) بأي وسيلة سيتم إجراء العمليات الحسابية والى أي درجة من الدقة ؟ .
٣) ماهو تسلسل إجراء العمليات لحل تلك المشكلة ؟ .
٤) هل هناك وسيلة لحفظ النتائج الوسيطة والتي سيتطلب استخدامها في مراحل لاحقة للحصول في النهاية على الحل ؟ .
٥) ماهي النتائج المطلوب الحصول عليها تمثل حلا للمشكلة ؟ .

ولنبدأ بالأجابة عن التساؤلين الأول والثالث . فمن العلاقتين السابقتين يتضح بوضوح عدم إمكانية حساب قيمة Y قبل حساب قيمة X . ولكن لحساب قيمة X ، فلا بد من معرفة قيم C,B,A وكذلك لحساب قيمة Y فيجب أيضاً معرفة قيمة D .

أي أننا لحساب قيمة Y لابد أولاً من معرفة قيم المتغيرات D, C, B, A وهذا ما يجيب على التساؤل الأول . ومما سبق يتبين أيضا أن تسلسل إجراء العمليات سيكون كالتالي :

١) معرفة قيم D, C, B, A
٢) حساب قيمة X من العلاقة : $X = A + B - C$
٣) حساب قيمة Y من العلاقة :

$$Y = \frac{X}{D}$$

٤) كتابة قيمة Y المطلوب الحصول عليها .

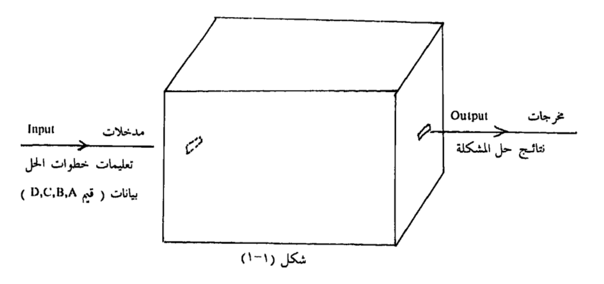
كذلك يتضح أن تلك المشكلة تتطلب اجراء بعض العمليات الحسابية الكلاسيكية مثل الجمع والطرح والقسمة ، ولحسن الحظ أنها لا تحتوي على عمليات حسابية خاصة مثل ايجاد جذر تربيعي

أو لو غاريتم عدد ما أو حساب قيم مرفوعة الى قوى أكبر أو أصغر من الواحد أو دوال مثلثية أو غير ذلك . ولذا فقد يتطلب الأمر إجراء مثل تلك الحسابات في مشكلة أخرى ، وبالتالي فلابد من تواجد وسيلة تستخدم لعمل تلك الحسابات سواء كانت تلك الوسيلة يدوية أو آلات بدائية . وفي الحالتين لابد من تدخل العنصر البشري وما يستتبع ذلك من تفاوت في **السرعة والدقة** ، ولعل ذلك يجيب على التساؤل الثاني .

وحيث أننا افترضنا أن الأدوات التي لدينا لاتتعدى الورقة والقلم ، فأنه من البديهي أن تكون الورقة وما عليها من حسابات هي الوسيلة الوحيدة لحفظ (تخزين) البيانات والنتائج الوسيطة (قيمة X وقيم المتغيرات D, C, B, A في مثالنا الحالي) . وفي هذه الحالة فأن **سرعة** الحصول على عمليات حسابية وسيطة سبق تدوينها ويراد استخدامها فيما بعد ، ستتوقف على العنصر البشري أيضاً وطريقة تنظيمه في وضع تلك الحسابات الوسيطة على سطح الورقة .

وأخيراً لابد لنا أن نتساءل ماهو الهدف الذي نريد أن نصل اليه بعد إجراء العمليات الحسابية الوسيطة وتجميعها والتعويض بها في العلاقتين السابقتين ؟ والاجابة بديهية ، وهي الحصول على قيمة Y بدرجة تقريب تتوقف على العمليات الحسابية الوسيطة والنهائية . ولذا فان العمليات الحسابية الوسيطة (حساب قيمة X) لم تكن الا مرحلة وجب **تخزينها** والاحتفاظ بها لاستخدامها في الوصول الى الهدف النهائي وهو حساب قيمة Y .

والآن لنتخيل مرة أخرى جهازا على شكل صندوق كبير لا نعرف ماذا يحوي بداخله ولا يظهر منه سوى فتحتين متقابلتين الشكل (١-١) . وسنتخيل أن أحدى الفتحتين ستكون مخصصة لادخال قيم D, C, B, A **بطريقة ما** ، وكذلك خطوات حل المشكلة .



شكل (١-١)

والفتحة الأخرى للحصول على نتيجة قيمة Y **بطريقة ما أيضاً** . أي على النتيجة النهائية للمشكلة ، فماذا يمكن أن نتوقع وجوده بداخل ذلك الصندوق ؟

وللا جابة على ذلك التساؤل سنجد أن هذا الصندوق لابد وأن تكون له القدرة على :

١) ترجمة وتنفيذ خطوات حل المشكلة بالترتيب الذي أدخلت له ،

٢) تخزين بيانات المشكلة وخطوات حلها وكذلك النتائج الوسيطة التي تلزم للحصول على النتائج النهائية .

٣) القيام بتنفيذ العمليات الحسابية التي تتطلبها حل المشكلة .

وفي الواقع اذا أضفنا للثلاث قدرات السابقة إمكانية :

٤) ادخال تعليمات خطوات الحل والبيانات الخاصة بالمشكلة .

٥) إخراج النتائج النهائية والوسيطة (ان تطلب الأمر ذلك) .

لحصلنا على حاسب عددي تتوقف قدراته على مدى كفاءة الخمس وحدات أو قدرات السابق ذكرها . ومما سبق يمكننا أن نستخلص أن **الوحدات الأساسية المكونة لأي حاسب عددي هي :**

١) وحدة لأدخال التعليمات والبيانات **Input Data Unit**

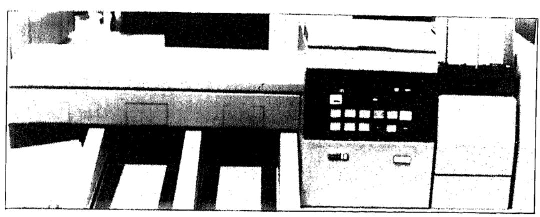
وتكون مهمتها قراءة التعليمات والبيانات . ونقصد هنا بالتعلميات ، خطوات الحل مرتبة في تسلسل منطقي في صورة أوامر أو تعليمات Statements or Instructions .

أما البيانات Data فنقصد بها القيم العددية التي تلزم لحل المشكلة . ومن هذا يتضح أن التعليمات والبيانات ستكون مزيجا من الحروف Letters والأعداد Numbers والعلامات الخاصة Special characters (والتي سنذكرها فيما بعد) .

وتوجد في وقتنا الحاضر وسائل عديدة يمكن عن طريقها ادخال التعليمات والبيانات الى الحاسبات العددية منها :

(أ) وحدة قراءة البطاقات المثقبة **Card Reader Unit**

وكانت تعتبر حتى أواخر الستينات من أهم الطرق وأكثرها شيوعاً . وفي هذه الطريقة يتم وضع التعليمات والبيانات على بطاقة ورقية ذات سُمك وملمس خاص وفي صورة بحيث أن كل رقم أو حرف أو علامة خاصة تمثل بثقب أو بمجموعة رأسية من الثقوب تميز ذلك الحرف أو الرقم أو العلامة عن غيره من الحروف والأرقام والعلامات . وتتم قراءة تلك البطاقات المثقبة عن طريق معرفة الثقب أو مجموعة الثقوب الموجودة في المواضع المختلفة من البطاقات وترجمتها الى نبضات كهربية Electric pulses تمثل الحرف أو الرقم الذي تم قراءته ، (الشكل ١-٢) .



شكل (١-٢)

وقد بدأ الاعتماد على تلك الطريقة يقل تدريجياً بظهور طرق تكنولوجية أخرى حديثة مثل وحدات القراءة عن طريق الأشرطة المغناطيسية أو الاسطوانات أو الأقراص الممغنطة وجميع تلك الطرق الحديثة تمتاز في سرعة ادخال البيانات للحاسب وفي حجم البيانات التي يمكن أن تستوعبها على مساحات صغيرة منها .

(ب) وحدة قراءة الشرائط الورقية المثقبة : Punched Paper tape reader unit:

وتشبه الوحدات السابقة ، الا أن التعليمات والبيانات يتم تثقيبها على شريط ورقي ، كما أن طريقة قراءة تلك الثقوب تشبه تلك المستخدمة في البطاقات المثقبة .

(جـ) وحدة قراءة الشرائط والأقراص الممغنطة :

Magnetic tape or disc reader unit:

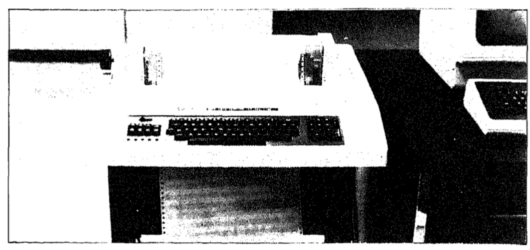
ويتم تسجيل البيانات والتعليمات على أشرطة أو اسطوانات ممغنطة شبيهة بتلك التي تستخدم في أجهزة المسجلات Tape Recorders ثم تغذى بها وحدة القراءة .

(د) وحدة ادخال البيانات والتعليمات عن طريق الوحدات الطرفية :

The Teletype devices:

ويعتبر إحدى الوسائل الشائعة الاستخدام في ادخال البيانات في الحاسبات وتمتاز بسهولة استخدامها وإمكانية تغذية الحاسبات بالبيانات والتعليمات عن طريقها من مسافات بعيدة (الشكل ١-٣) .

ولذا فكثيرا ما تستخدم مثل هذه الوحدات في الأغراض التجارية مثل شركات الطيران والبنوك والمؤسسات الكبرى حيث ترتبط جميعها بالحاسب عن طريق توصيلات تليفونية وسبب سهولةاستخدامها هو في أنها تشبه الالآت الكاتبة Typewriters في إستخدامها كما أن البيانات والتعليمات



شكل (١-٣)

التي تعطى للحاسب عن طريقها تظهر مكتوبة واضحة بحيث تقلل من فرصة دخول أرقام أو حروف غير صحيحة للحاسب .

(هـ) وحدة ادخال البيانات عن طريق الوحدات الطرفية ذات الشاشة :

Cathode Ray Tube Screens or Video Devices:

وتعتبر هذه الوحدات استحداثا للوحدة السابقة د ، حيث أن البيانات والتعليمات التي يغذى بها الحاسب تظهر في نفس الوقت على شاشة تليفزيونية مضيئة .



شكل (١-٤)

وبجوار هذه الوحدات توجد وسائل أخرى أقل استخداما وشيوعا مثل :
الكاسيتات المغنطة Magnetic cassets ، الحبر المغنط Magnetic Ink ، الاسطوانات المغنطة Magnetic discs ؛ البطاقات المغنطة Magnetic cards ، وكذلك عن طريق الاتصال الصوتيVoice communication . التي تستخدم في نطاق ضيق ومحدود .

٢) وحدة استخراج النتائج : Output Unit

وتتركز مهمتها في إستخراج النتائج الوسيطة والنهائية ، بجانب كتابة البيانات والمعلومات التي سبق قراءتها عن طريق وحدة الادخال . وأهم وحدات استخراج النتائج المعروفة في وقتنا الحاضر هي :

الطابعات : Line Printers

وتعتبر وسيلة الاستخراج الأكثر شيوعاً ولا يخلو منها أي حاسب عددي ، فعن طريقها يمكن الحصول على النتائج مطبوعة على الورق بحيث يمكن استخدامها مباشرة بخلاف بعض وحدات الاستخراج الأخرى والتي قد تتطلب أجهزة مساعدة لترجمة المستخرجات كي تظهر بالصورة التي يمكننا أن نقرأها بها ، كما يعتبرالطابع أحد الوسائل السريعة والسهلة الاستخدام .

ويمكن اعتبار وحدات الادخال والتي سبق أن أشرنا اليها كوحدات استخراج أيضا .

٣) وحدة التخزين : Storage or Memory Unit

وفيها يتم تخزين جميع البيانات والتعليمات التي تعطى للحاسب عن طريق وحدات الأدخال وكذلك الأوامر الخاصة بحل المشكلة بترتيبها المنطقي ، وأخيراً نتائج العمليات الحسابية الخاصة بالمشكلة وبالتالي النتائج النهائية لها .

وتعتبر وحدات التخزين في الحاسبات أحدى عوامل تفضيل ما بين حاسب وآخر من ناحيتين :

(أ) كمية البيانات والتعليمات التي يمكن أن تستوعبها والتي تعرف علمياً بسعة التخزين Storage capacity .

(ب) سرعة تجاوب وحدة التخزين والتي تقاس بعاملين :

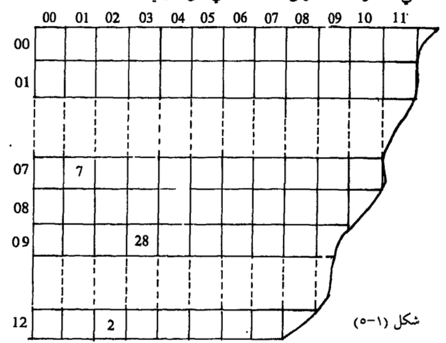
١) الزمن اللازم لتخزين معلومة .

٢) الزمن اللازم لقراءة تلك المعلومة أي الزمن الذي تأخذه وحدة التخزين في البحث عن موضع معلومة ما بداخلها .

ومجموع الزمنين السابقيين هو ما يطلق عليه بأسم زمن التجاوب Access time وبالتالي فكلما كانت سعة وحدة التخزين لحاسب ما كبيرة وزادت سرعة تجاوبها (أي كان زمن التجاوب بها قصيرا) كلما أرتفعت كفاءة ذلك الحاسب وكان أفضل من غيره من الحاسبات .

ووحدات التخزين في الحاسبات الألكترونية الحديثة تتركب من حلقات ممغنطة Magnetic cores أو من داوئر متكاملة Integrated circuits وكل موضع في وحدة التخزين يمكن التعرف عليه

عن طريق رقم يرمز له ويطلق عليه إسم عنوان Address . ويجب هنا أن نفرق بين عنوان معلومة ما وبين مايوجد من بيانات في داخل هذا العنوان . ففي المثال السابق إذا فرضنا أن قيمة A هي 7 ,B هي 28 ,C هي 2 ، وكانت عناوين C, B, A هي على التوالي :



شكل (١-٥)

1202, 0903, 0701 فأننا نقول مثلاً أن بداخل الموضع الذي عنوانه 0701 (وهو يمثل موضع A) توجد القيمة 7 أي أن قيمة A هي 7 . ويقوم الحاسب بوضع عنوان لكل متغير يتم تخزينه ، مالم يعط أمرا لتخزين متغير ما في موضع معين بوحدة التخزين .

وبعض الحاسبات وخاصة الكبيرة منها تحتوي على أكثر من وحدة تخزين احداهما أساسية أو مركزية Main Storage or Central memory ولا بد من تواجدها في أي حاسب وتتميز بسرعة تجاوبها وسعتها المحدودة نسبياً . أما بقية وحدات التخزين (ان وجدت) فيطلق عليها وحدات تخزين ثانوية Auxiliary Secondary memory وهي تتميز بسعة تخزين كبيرة ولكن بسرعة تجاوب أقل نسبياً من الأولى .

٤) وحدة الحساب والمنطق : The Arithmetic Logic Unit

وهي الوحدات التي تقوم بتنفيذ العمليات الحسابية العددية والمنطقية على البيانات التي سبق تخزينها في الحاسب وطبقاً لتسلسل تنفيذها في مجموعة التعليمات والأوامر والتي تكون ما يعرف باسم البرنامج The Program (والذي سنتطرق الى شرحه بالتفصيل في الباب الثاني). فلاجراء

عملية حسابية على متغيرين فانه يتم تخزين قيمتى هذين المتغيرين في وحدات تسجيل Registers موجودة داخل تلك الوحدة والتي يتم في احداها تسجيل نتيجة تلك العملية تمهيداً لتخزينها في المكان المخصص لها بوحدة التخزين الرئيسية للحاسب .

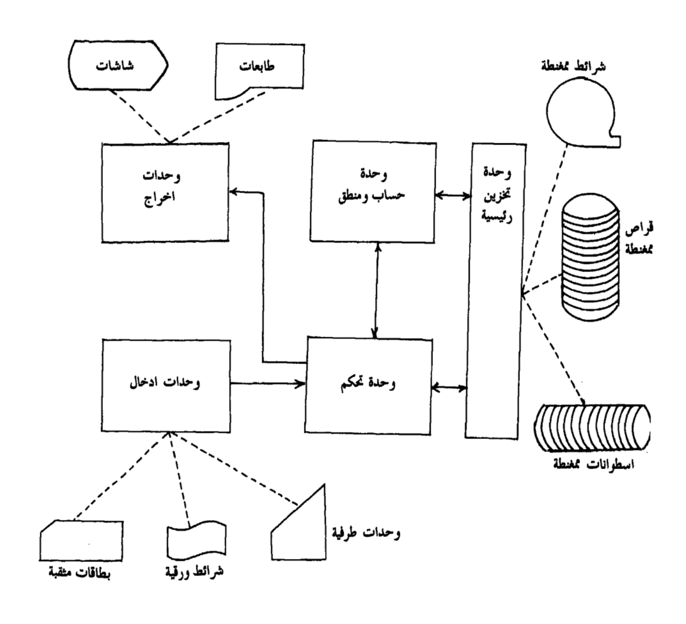
ويلاحظ من إسم تلك الوحدة أنها تقوم بتنفيذ العمليات الحسابية العددية مثل الجمع والطرح والضرب ...الخ وكذلك العمليات المنطقية Logical Instructions مثل المقارنة بين قيميتين من ناحية الكبر أو الصغر أو التساوي أو عدم التساوي أو عمليات منطقية أخرى سنتعرض لها في حينها .

وكما لوحدة التخزين من أهمية في المفاضلة بين حاسب وآخر فكذلك الوحدة الحسابية ، حيث تقاس كفاءة الحاسب بالنسبة لهذه الوحدة بمدى تجاوب وسرعة تنفيذ العمليات الحسابية والمنطقية . وقد نلمس مدى الفارق في التطور الذي حدث في تلك الوحدة بالذات اذا تذكرنا حاسبات الجيلين الأول والثاني والتي كانت تعتمد في تنفيذ العمليات الحسابية على تعشيق تروس وادارتها وما يتطلبه ذلك من زمن ومجهود أو في إستخدام الصمامات وما كان يتطلبه ذلك من تجهيزات أخرى في التغلب على الحرارة الكبيرة المنبعثة منها والرقابة والصيانة المستمرتين على كفاءة عمل تلك الصمامات بجانب ارتفاع اثمانها بالمقارنة باستخدام الدوائر المتكاملة أو الترانزستورات المستخدمة حالياً في تركيب تلك الوحدات .

٥) وحدة الرقابة أو التحكم : The Control Unit

وتعتبر هذه الوحدة من أنشط الوحدات السابقة فهي في عمل دائم ومستمر طالما أن الحاسب يقوم بتنفيذ عملية حسابية ما ، اذ تشتمل هذه الوحدة على الدوائر التي تتحكم في وضع البيانات والتعليمات داخل وحدة التخزين الرئيسية للحاسب وذلك بترتيبها وترجمة كل منها ، وكذلك التعامل مع الوحدة الحسابية والوحدات الأخرى لاجراء وتنفيذ تلك التعليمات . وتقوم وحدة التحكم بتخزين التعليمات والأوامر التي يحتويها البرنامج في أماكن محددة من وحدة التخزين (معروفة عناوينها) كما تقوم بتخزين البيانات الخاصة بالمشكلة في أماكن أخرى ثم تبدأ وحدة التحكم في تنفيذ تلك التعليمات حسب تسلسلها في البرنامج الذي يراد الحساب عليه وعند تنفيذ تعليمة ما تقوم وحدة التحكم بالتعرف على نوع العملية المراد تنفيذها (جمع أم طرح ...الخ) وذلك بمعرفة شفرة العملية Operation Code والبيانات التي تلزم لتنفيذها ونقل تلك الشفرة والبيانات الى وحدات التسجيل في وحدة الحساب والمنطق لتنفيذها ثم تقوم تلك الوحدة بنقل النتيجة الى المكان المخصص لها في وحدة التخزين .

ولذا فانه يتبين مما سبق أن هناك ترابطاً تبادلياً بين وحدة التحكم وكل من وحدة الحساب والمنطق ووحدة التخزين . وعموماً فأنه يمكن تمثيل الترابط بين الوحدات المختلفة السابقة وبين بعضها وكذلك وحدات الأدخال والاخراج ووحدات التخزين الثانوية كما هو مبين في (الشكل ١-٦) .



شكل (١-٦)

١-٣ البيانات اللازمة لحل مشكلة ما على الحاسبات الألكترونية :

عند بدء ظهور الحاسبات الألكترونية في العالم العربي أطلق البعض عليها اسم « العقول الألكترونية » ولا شك أن هناك فارقا كبيراً بين الحاسب والعقل . فبدون أن يمهد العقل للحاسب طريقة الحساب والبيانات اللازمة للمشكلة فلن يستطيع الحاسب أن ينفذ عملية حسابية واحدة ولو كانت من أبسطها . وكل ما يمتاز به الحاسب عن العقل البشري هو سرعة انجاز العمليات الحسابية والدقة في حسابها ، ففي الزمن الذي يستغرقه العقل البشري في عملية جمع عددين بدرجة تقريب معينة تستطيع بعض الحاسبات تنفيذ أكثر من مليون عملية جمع لأعداد تحتوي على ٣٢ أو ٦٠ رقماً .

وسبق أن عرفنا أن هناك وحدتين من الوحدات الأساسية للحاسبات الألكترونية هما وحدتى ادخال البيانات والتعليمات – ووحدة اخراج النتائج . ولذا فلابد لنا أن نجهز أنفسنا بتحضير ما سيتم اعطاؤه للحاسب من بيانات وتعليمات عن طريق وحدة الادخال ، وكذلك لابد لنا من التفكير فيما نريد إستخراجه من نتائج للمشكلة ، هل نكتفي باستخراج النتائج النهائية للمشكلة ؟ أم يفضل استخراج نتائج وسيطة ، خاصة للعمليات الحسابية التي تتطلب وقتاً كبيراً وذلك للتأكد من صحة تلك النتائج النهائية .

وإذا تأملنا في أي مشكلة حسابية نجد أنها لابد من أن تمر بمراحل ثلاث لحلها :

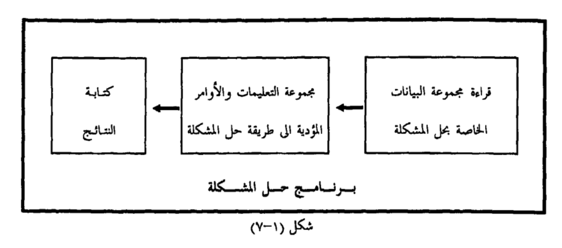
١) توفر البيانات اللازمة لحل المشكلة .

٢) طريقة الحل (الخوارزم(*)) The Algorithm . أو مجموعة التعليمات والأوامر التي لابد للحاسب من تنفيذها .

٣) تحديد النتائج المطلوب إستخراجها .

وهذه المراحل الثلاث مجتمعة تكون فيما بينها اسم البرنامج The Program وهي موضحة في (الشكل ١–٧) .

(*) الخوارزم : كلمة غربية تطلق على مجموعة خطوات تنفذ بطريقة تسلسلية بهدف الحصول على الحل النهائي . وهذه الكلمة اطلقها الغربيون إعترافاً وتقديراً للعالم المسلم محمد إبن موسى الخوارزمي الذي عاش في بغداد بين عام ١٦٤هـ وعام ٢٣٥هـ وتوفي هناك . وبرز في علم الرياضيات والفلك في عهد الخليفة المأمون . كما خلد أيضاً بابتكاره علم حساب «اللوغاريتمات» .



شكل (١-٧)

مثال (١) :

باستخدام الحاسب ، أوجد جذري معادلة الدرجة الثانية : $AX^2 + BX + C = O$

أولاً : يتطلب لايجاد جذري المعادلة السابقة معرفة قيم C, B, A والتي يمكن اعتبارها أنها البيانات التي يجب توفرها .

ثانياً : اذا علمنا أن جذري معادلة الدرجة الثانية يعطى من القانونين :

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad ; \quad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

فلا يتبقى الا أن نعطي الحاسب مجموعة التعليمات والأوامر بحساب قيمتى X_1, X_2 . وطريقة الحل قد تختلف بين شخص وآخر ، فقد تكون مجموعة تعليمات وأوامر بالتعويض المباشر بقيم C, B, A في القانونين السابقين بغض النظر عما اذا كانت قيمة A تساوي صفرا أم لا ، أو أن تكون قيمة $(B_2 - 4AC)$ سالبة ، والذي قد يتسبب عنها توقف الحاسب عن تنفيذ بقية البرنامج أو اعطاء نتائج مضللة .. ولهذا فأنه يجب علينا مراعاة الحالات التالية والتي سيترتب عليها توقف البرنامج في حالة حدوث أحدها كي نقوم بمعالجتها من خلال التعليمات المناسبة .

الحالة الأولى :

عندما تكون قيمة A تساوي صفرا فإن المعادلة تؤول الى معادلة من الدرجة الأولى $BX + C = O$ ويكون هناك جذر واحد هو $X = - C/B$.

الحالة الثانية :

عندما تكون $(B^2 - 4AC)$ سالبة ، ستكون الجذور مركبة وفي هذه الحالة يجب أن يحتوي الخوارزم على طريقة لحسابها .

الحالة الثالثة :

تكون $(B^2 - 4AC)$ غير سالبة وقيمة A تختلف عن الصفر عندئذ لن تكون هناك مشاكل في حساب قيمتى X_1 ، X_2 وكتابة قيمهما بعد أن يقوم الحاسب بالتعويض في القانونين السابقين .

فأذا اردنا المفاضلة بين مجموعتى التعليمات والأوامر سنجد أن الأخيرة تفوق الأولى في أنها تحوي جميع الاحتمالات التي قد يصادفها الحاسب عند تنفيذ البرنامج وما يجب عليه عمله عند ظهور أي منها ، لأننا كما ذكرنا أن الحاسب عديم القدرة على التفكير ولكنه سريع التنفيذ طالما أن التعليمات والأوامر واضحة ومحددة له .

مثال (٢) :

بأستخدام الحاسب ، أوجد قيمة كثيرة الحدود المعطاه بالمعادلة :

$$Y = 2X^3 + 3X^2 + 5X + 3 \qquad (1)$$

في هذه الحالة سنجد أنه لحساب قيمة كثيرة الحدود ، أي لايجاد قيمة Y :

أولاً : يتطلب معرفة قيمة X ، حيث أنه عند كل قيمة للمتغير X سنجد أن Y ستكون لها قيمة . وإذا كانت كثيرة الحدود السابقة في الصورة :

$$Y = AX^3 + BX^2 + CX + D \qquad (2)$$

فأنه في هذه الحالة يلزمنا معرفة قيم D, C, B, A بجانب قيمة X حتى يمكننا في النهاية حساب قيمة المتغير Y . وسنفترض في مثالنا الحالي كثيرة الحدود المعطاه بالمعادلة (١) والتي يلزم لحسابها معرفة قيمة X فقط والتي يجب اعطاؤها للحاسب اذا أريد معرفة قيمة Y عند القيمة المعطاه للمتغير X .

ثانيـاً : اذا أعطينا الحاسب قيمة X فأن حساب قيمة Y يمكن الحصول عليها بأحدى طريقتين :

(أ) بأعطاء الحاسب مجموعة التعلميات والأوامر بحيث يقوم بحساب $5X$ ، $3X^2$, $2X^3$ ويقوم بجمعها ويضيف الى الناتج القيمة 3 . وفي النهاية نعطيه الأمر بكتابة قيمة Y .

(ب) بتطوير كثيرة الحدود أولا وجعلها في الصورة :

$$Y = ((2X + 3)X + 5)X + 3 \qquad (3)$$

وهي نفس المعادلة (١) جبريا بعد فك الأقواس .

فأذا أردنا المقارنة بين الطريقتين سنجد أن الطريقة الثانية (ب) ستكون أسرع واكثر دقة حيث أنها لن تتطلب من الحاسب اكثر من عمليات ضرب وجمع بينما الطريقة الأولى (أ) ستستلزم حساب قوى مختلفة للمتغير X والتي ستتطلب من الحاسب زمنا أطول(*) وستكون النتيجة أقل دقة من عمليات الضرب فقط .

ثالثاً من الواضح أن النتائج المطلوب استخراجها هي قيم Y عند اعطاء قيم X ، ولذا فقد يكون من الأنسب في هذا المثال إعطاء الحاسب الأمر بكتابة قيمة X المعطاه وقيمة Y المناظر لها .

ونستخلص مما سبق أنه لحل مشكلة ما على الحاسبات الألكترونية فأنه يلزم :

١ – اعطاء الحاسب مجموعة التعليمات والأوامر التي تمثل برنامج حل المشكلة والذي يحتوي على :

(أ) أوامر قراءة البيانات اللازمة لحل المشكلة .

(ب) طريقة حل المشكلة ، واذا كان هناك اكثر من طريقة فأن درجة التفضيل تكون لتلك الأكثر دقة والأقل زمناً (إن أمكن) في الحصول على النتيجة .

(جـ) أوامر لكتابة النتائج سواء النهائية أم الوسيطة كذلك .

٢ – أن يشمل البرنامج – قدر الأمكان – على مختلف الاحتمالات بحيث يوضح للحاسب ما يجب عمله عند التعرض لتلك الاحتمالات .

٣ – في معظم الأحيان ، وخاصة عند حل مشاكل كبيرة تتطلب مجموعة كبيرة من الأوامر والتعليمات ، فأنه من المفضل جعل الحاسب يكتب نتائج وسيطة عند مواضع مختلفة من البرنامج بحيث يكون من السهل معرفة موضع بداية الخطأ في الحسابات ، إذا ظهر أن النتيجة النهائية غير صحيحة وهذا يوفر كثيرا من الوقت والجهد في البحث من أول البرنامج الى موضع ذلك الخطأ ، وسنلمس ذلك عندما نتمكن من كتابة برامج تشمل عشرات أو مئات الأوامر والتعليمات .

٤ – أن نضع في أذهاننا والا يغيب عنا ، أن الحاسب ماهو الا أداة حسابية منفذة بكل دقة وبدون تفكير ، للأوامر والتعليمات التي يتلقاها ، وكلما كانت تلك الأوامر

(*) تحوي الحاسبات بعض البرامج الجزئية Subprograms لتبين للحاسب طريقة حل بعض الدوال الخاصة مثل ايجاد الجذر التربيعي لقيمة ما وحساب لوغاريتم عدد ما أو قيمته المطلقة أو ... الخ . وعلى سبيل المثال فأنه لحساب قيمة متغير مرفوع الى قوة ما فان ذلك يتطلب حساب متسلسلة أسية Power Series متقاربة يلزم لحسابها عمليات ضرب تفوق كثيراً العمليات التي تلزم لها الطريقة (ب) ، كما أن قيمة تلك المتسلسلة ستكون تقريبية (يمكن الرجوع في ذلك الى كتب التحليل العددي) .

والتعليمات واضحة وشاملة ومطابقة لقواعد اللغة التي يتعامل بها ، كلما أمكننا توفير الجهد والوقت والتوصل الى النتيجة الصحيحة .

١-٤ المراحل المختلفة التي يمر بها برنامج على الحاسبات الألكترونية :

كلنا يتقن اللغة العربية قراءة وفهما وكتابة ونلم الماماً تاماً بفهم قواعدها ، ولذا لانواجه بأية عقبات عند قراءتها أو فهمها أو كتابتها . والبعض منا لايتقن الأنجليزية مثلا ، وبالتالي سيواجه بعقبات وصعوبات عند قراءتها أو كتابتها لأنه لايعرف قواعد تلك اللغة المعرفة الكافية والتي قد تسبب له الوقوع في أخطاء كثيرة وربما في التوقف كلية عن القراءة أو الكتابة . ولكن إذا أعطينا قاموسا لمساعدتنا في قراءة ومعرفة قواعد تلك اللغة فلن تكون هناك مشكلة أو صعوبة في قراءتها أو كتابتها ، وينطبق نفس الشيىء على أية لغة أخرى لانجيدها .

وكما سبق أن علمنا أن الحاسبات ماهي الا أدوات تنفيذ لأوامر وتعليمات بسرعة كبيرة طالما أن تلك الأوامر والتعليمات واضحة لها ومطابقة لقواعدها . فأذا اردنا أن نتعامل مع تلك الحاسبات فما علينا الا أن نعطيها أوامر وتعليمات وبيانات المشكلة التي نريد حلها باللغة التي تجيدها ، الا وهي لغة الحاسب نفسه Machine Language . ونظراً لأن الحاسبات تعمل بدوائر الكترونية Electronic Switching Circuits يتحكم فيها عنصرى حساب النظام الثنائي The Binary System أي الصفر والواحد ، ولذا فأن اللغة المباشرة والتي يمكن التعامل بها مع الحاسبات هي تلك التي تحتوي على هذين العنصرين فقط . ونظراً للصعوبة الكبيرة التي ستواجهنا في التعامل مع الحاسب بلغته ، فقد أمكن عمل مايشبه قواميس للغات مختلفة يمكننا الكتابة بها ويقوم الحاسب بمساعدة تلك القواميس الى ترجمتها الى لغته . ويطلق على تلك القواميس أو الترجمات اسم المصنفات أو المجمعات Compilers .

ولذا فلكي يتمكن الحاسب من فهم لغة ما غير لغته لابد أن يحولها **أولاً** الى لغته . ولكي ينفذ التعليمات المعطاه له في صورة برنامج فلابد أن تعطى تلك التعليمات للحاسب بشرط أن تكون متفقة مع قواعد اللغة التي ترجمها الى لغته ومن هنا تأتي أهمية دراسة القواعد والشروط التي لابد من اتباعها لكتابة برنامج بلغة غير لغة الحاسب .

وفي بدء ظهور الحاسبات كانت هناك لغة واحدة معروفة قريبة من لغة الحاسب تسمى Assembly Language وكانت تترجم الى لغة الحاسب باستخدام برنامج وسيط يسمى Assembler . ونظراً لصعوبة تلك اللغة فقد استحدثت لغات أخرى بعضها يفوق الآخر في تطبيقات علمية معينة وعددها الآن يزيد عن خمسمائة لغة أهمها وأكثرها شهرة واستخداما هي اللغات :

FORTRAN - BASIC - COBOL - ALGOL - PASCAL - PL1.

وكل لغة من تلك اللغات لها قواعدها وامكانيات تطبيقها في مجالات علمية وعملية معينة ، فعلى سبيل المثال فأن اكثر مجالات استخدام لغة الفورتران FORTRAN هي في الحسابات العددية والهندسية ، بينما تستعمل لغة الكوبول COBOL غالبا في الحسابات التجارية . وهذا لايمنع استخدام اللغات الباقية في المجالين السابقين ولكن سيكون ذلك بامكانيات وكفاءة تقل عن اللغتين السابقتين . ويطلق على تلك اللغات اسم اللغات ذات المستوى العالي High level language .

ولذا فقبل البدء في تنفيذ برنامج بلغة ما على الحاسب لابد من التأكد أولا أن الحاسب يحوي في ذاكرته (وحدة التخزين) المصنف الخاص بتلك اللغة والا فأن الحاسب لن يستطيع فهمها وبالتالي ترجمتها وتنفيذها . وفي عصرنا الحالي فأن الحاسبات الكبيرة يمكنها التعامل مع أكثر من لغة في وقت واحد نظراً لأن مصنفات تلك اللغات يكون قد سبق وضعها في وحدات تخزين تلك الحاسبات .

وعند حل مشكلة ما على الحاسب فأنه لابد من المرور بالمراحل التالية :

١ - تحضير خطوات حل المشكلة بعمل رسم تخطيطي لها يبين انسياب تلك الخطوات ومدى تسلسلها وترابطها مع بعضها البعض . ويطلق على هذا الرسم التخطيطي اسم « مخطط تدفق الخطوات الحسابية والمنطقية » FLOW CHART . وسيأتي الحديث عنها بالتفصيل في الفصل الثاني .

٢ - كتابة برنامج حل المشكلة (بالأستعانة بمخطط التدفق اذا تطلب الأمر ذلك) بلغة يكون قد سبق التأكد من أن الحاسب يتعامل معها ويستطيع ترجمتها (وسنقصر حديثنا في هذا الكتاب على لغة الفورتران وقواعدها) .

٣ - يغذى الحاسب بالبرنامج عن طريق وحدة من وحدات ادخال التعليمات والبيانات .

٤ - يقوم الحاسب باختبار مدى مطابقة التعليمات الواردة في البرنامج بقواعد اللغة التي كتب بها ، واذا كانت هناك اخطاء فأن الحاسب يكتشفها ويجب تصحيحها أولا قبل البدء في ترجمة تلك التعليمات .

٥ - بعد تأكد الحاسب من أن البرنامج لايحتوي على اخطاء خاصة بقواعد اللغة التي كتب بها يبدأ في ترجمة البرنامج الى لغته (أي لغة الحاسب) . وتسمى المرحلتين ٥،٤ بعملية التصنيف Compilation والتي تشمل بداخلها عمليات :
الترجمة Translation والتجميع Assembly وتكوين التركيب Structuring .

٦ – بعد ذلك تبدأ مرحلة التنفيذ Execution ويقوم فيها الحاسب بتنفيذ الأوامر بالترتيب الذي كتبت به في البرنامج بعد أن تكون قد ترجمت الى لغة الحاسب .

٧ – في مرحلة التنفيذ ، قد يكتشف الحاسب نوعا آخر من الأخطاء أثناء تنفيذ بعض العمليات الحسابية مثل عملية قسمة تحوي في مقامها صفرا أو كمية سالبة يراد حساب جذرها التربيعي لها أو لوغاريتم لكمية سالبة وهكذا . وبعض الحاسبات تعطي تحذيرا مكتوبا على وحدة استخراج النتائج المستخدمة عند اكتشافها لمثل هذا النوع من الأخطاء وقد يظهر ذلك التحذير أيضا في صورة ضوئية (إضاءة لمبة) على وحدة التحكم ، وقد يتوقف تنفيذ البرنامج بعد ذلك لأصلاح الخطأ أو يستمر في تنفيذ باقي تعليمات البرنامج ولكن بنتائج وسيطة مشكوك فيها . ولذا فأنه يفضل قبل تنفيذ برنامج على الحاسب أن يتم حساب المشكلة يدويا بمثال بسيط ويتم تنفيذ البرنامج مستخدمين نفس البيانات البسيطة فأن تطابقت النتيجتين أمكن إستخدام البرنامج بإطمئنان لحل مشاكل كبيرة .

الفصـــل الثـاني

الثوابت والمتغيرات والعمليات الحسابية

الفصــل الثـاني

مـقـدمــة :

قد يتساءل القارىء لأول وهلة بعد الانتهاء من قراءة الفصل الأول عن الفارق بين الآلة الحسابية The Calculator والحاسب الآلي The Computer ؟ . وقد يقول قائل لماذا يتجشم عناء استعمال الحاسب الآلي ونخوض في تعقيدات لاطائل منها طالما أن الآلة الحاسبة تستطيع القيام بنفس المهام وبالتالي فأننا نستطيع توفير كثير من الجهد والوقت والمال حيث أن الآلة الحاسبة لاتكلف الا مبلغا يسيراً من المال وبالتالي فهي في متناول الجيمع . وفي الواقع فأن كل هذه الأسئلة وغيرها تعتبر أسئلة منطقية ومعقولة ولكن هذا التساؤل يتلاشى ويذوب عندما نستعرض النقاط التالية .

أولاً : أن جزءا كبيرا من استعمال الآلة الحاسبة يتم بطريقة ميكانيكية يدوية عن طريق الضغط على الأزارير المختلفة ، وبالتالي فأن على مستعمل الآلة الحاسبة أن يقوم بكل الخطوات العملية واحدة بعد الأخرى حتى ولو كانت هذه الخطوات مكررة ومملة وبنفس النمط والأسلوب بينما يقوم الحاسب الآلي بنفس المهمة بطريقة آلية بحتة تتم بواسطة تنفيذ بعض الأوامر والتعليمات الصادرة اليه عن طريق برنامج مكتوب بلغة خاصة وطريقة خاصة .

ثانياً : اذا افترضنا أن سرعة اجراء العمليات الحسابية مثل الجمع والطرح ... الخ في الالآت الحاسبه تتساوى مع نظيرتها في الحاسبات الآلية الا أنه من المؤكد أن درجة الدقة تختلف اختلافا كبيرا . فبعض المشاكل الفيزيائية والكيميائية تتطلب درجة كبيرة من الدقة قد تصل الى عدد من الأرقام العشرية لاتستطيع الالآت الحاسبة الوصول اليها بينما أصغر الحاسبات الآلية يمكن أن تتراوح نتيجة العملية الحسابية بها مابين $١٠^{-٣٨}$ ، $١٠^{٣٨}$. وعلى سبيل المثال فأن قيمة النسبة التقريبية في أي آلة حاسبة هي ٣٫١٤١٥٩٢٦٥٤ بينما يمكن حسابها بدقة أكبر بكثير في الحاسبات الآلية لتصل الى ٣٫١٤١٥٩٢٦٥٣٥٨٩٧٩٣٢٣٨٤٦٢٦٤٣٠٠٠ . وفي الحاسبات الآلية الكبيرة مثل CDC 6000/7000 يمكن أن تتراوح تنيجة العملية الحسابية مابين $١٠^{٣٢٢}$ ، $١٠^{-٣٢٢}$.

ثالثاً : أن لدى الحاسب الآلي المقدرة على التكرار Repetition . فهو لديه المقدرة على أن يقوم بالالآف بل مئات الألوف من العمليات المكررة في ثوان معدودة بواسطة برنامج لايتجاوز العشرة أسطر مثلا في حين أننا نحتاج الى عشرات أو مئات الساعات لانجاز نفس المهمة بواسطة الآلة الحاسبة ، والمقصود هنا بالعملية المكررة أي العملية التي تجري لأكثر من رقم ومثال ذلك ايجاد الجذر

التربيعي لعشرة أرقام موجبة مختلفة ، فالعملية هي ايجاد الجذر التربيعي وهي مكررة أما الأرقام والنتائج فهي متغيرة بتغير الأرقام .

مثال (١) :

لنفترض أن لدينا قائمة تحتوي على عشرة الآف رقم ، وطلب منا أن نوجد مربع كل رقم في هذا البيان .

الحـــل

في الواقع هناك أكثر من حل لهذه المسألة الطويلة القصيرة ، ويمكن تلخيص الحلول المختلفة كالتالي :

١ - **الحل اليدوي** : وهذا الحل قد يستغرق اسبوعاً كاملاً خصوصاً إذا كانت الأرقام كبيرة وذات أجزاء عشرية كبيرة .

١ - **الحل باستعمال الآلة الحاسبة** : وهي تتلخص في إدخال كل رقم على حدة ثم ايجاد مربع الرقم وتسجيل الناتج بعد كل مرة وهذا يتطلب تكرار العملية عشرة الآف مرة مما يستغرق جزءاً كبيراً في الوقت والجهد .

٣ - **الحل باستخدام الحاسب الآلي** : وهذا يتم عن طريق كتابة برنامج بسيط بلغة الفورتران أو أي لغة مماثلة ، يتكون من أربع أو خمس تعليمات بسيطة يرفق معه البيان بالكامل لنحصل بعد ثوان قليلة على بيان كامل يحتوي على جميع الأرقام ومربعاتها مطبوعة بشكل مرتب وأنيق .

ومما سبق يتضح لنا أن الاعتماد على الحاسب الآلي أصبح من البديهيات التي يقوم عليها عنصر السرعة والتكنولوجيا الحديثة التي تسعى الى التقدم المادي للإنسان لتوفر له مزيدا من الراحة والرفاهية والترف ، أما كيف ينفذ هذه الأوامر ، فيتم بواسطة تقديم هذه التعليمات والأوامر اليه في شكل خاص منظم يسمى بالبرنامج Program .

تعريف

البرنامج : سلسلة من التعليمات المكتوبة بلغة خاصة وترتيب معين

ومما لاشك فيه أنه بمرور الوقت فسيدرك الطالب مدى القدرات الهائلة للحاسب الآلي مقارنة بالآلة الحاسبة فضلاً على أن بعض العمليات والمشاكل لا يمكن أن تحل بغير الحاسب الآلي ويستحيل حلها بواسطة الآلة الحاسبة العادية حيث يرجع ذلك الى قدرة الحاسب على تخزين كميات كبيرة من المعلومات واستخدامها فيما بعد بعكس الآلة الحاسبة ذات القدرات المحدودة البسيطة .

« تمـارين عـامـة »

١ – حاول أن تستفسر عن امكانيات الحاسب الموجودة في المعمل ، كأن تسأل مثلاً عن عدد عمليات الجمع أو الضرب التي يستطيع أن يؤديها الحاسب خلال ثانية واحدة ؟ قارن الاجابة التي ستحصل عليها بالوقت الذي تحتاجه الآلة الحاسبة ؟

٢–١ التخطيط لكتابة برنامـج :

عندما تجابه الأنسان مشكلة من مشاكل الحياة العملية المعقدة فأنه بلا شك يحتاج الى كثير من التخطيط والتروي والتركيز وتقدير جيمع الاحتمالات الممكنة ليضع لها الحلول المناسبة والتصورات الصحيحة ليعمل بموجبها ويسير على ضوئها وبالتالي ينجح في حل تلك المشكلة بالطريقة المناسبة .. ولحل مشكلة علمية بواسطة الحاسب الآلي يحتاج الأمر الى كتابة برنامـج يؤدي الغرض المطلوب ، وكتابة البرنامـج تحتاج الى دراسة وتخطيط وتركيز وتقدير لمختلف الاحتمالات لوضع الحلول المناسبة . ولهذا فأن المبرمـج يحتاج عادة الى عوامل مساعدة ليتمكن من كتابة البرنامـج المطلوب .

ومن أهم هذه العوامل المساعدة استعمال مايسمى FLOWCHART والذي يمكن ترجمة معناه الى مخطط التدفق للعمليات الحسابية والمنطقية والذي يمكن التفكير فيه كرسم تخطيطي يوضح تسلسل التعليمات التي سيتكون منها البرنامـج ليؤدي الغرض المطلوب . وكما أن للرسم المعماري رموز خاصة ترمز الى اجزاء معينة من المبنى فإن لمخطط التدفق ايضاً بعض الرموز المتفق عليها والتي يدل كل منها على أمر معين . انظر الشكل (٢–١) .

الرمـز	الـدلالـة	المعنى
	نقطة بداية أو نهاية Terminal point	ترمز الى بداية أو نهاية البرنـامـج
	ادخال بيـانـات Input	ترمز الى ادخال بيانات رقمية أو غيرهـا من البيـانـات .
	إخـراج بيـانـات Output	ترمز الى إخراج بيـانـات رقمية وغيرها .

▭	عملية حسابيـة Process / Arithmetic Operation	ترمز الى اجزاء عمليـة حسابية كالجمع والطرح ... الخ .
◇	عملية منطقيـة Decision / Logical Operation	ترمز الى عملية منطقية يُتخذ فيها القرار اللازم ليتحدد على ضوئه اتجاه انسياب البرنامج .

الشكل (٢-١) رموز مخطط التدفق الانسيابي

اما اتجاه العمليات فسوف يحدد بواسطة استعمال الأسهم والتي تساعد على توضيح المخطط وتبسيطه للقارىء أو المبرمـج .

مثال (٢) :

ارسم مخططاً تدفقياً يوضح سير العمليات الحسابية لبرنامـج يقوم بمعالجة بيان يحتوي على رقم واحد فقط ليوجد مربع ذلك الرقم ومن ثم يكتب الناتج

الحـل

لو فكرنا قليلاً في الخطوات العملية لحل مشكلة بسيطة كهذه فأننا بلاشك سنصل الى اجماع عام بأن هذه الخطوات يجب أن تكون على الشكل التالي :

١ – نبدأ بقراءة الرقم المعطى (ادخال بيان) ولنرمز له بالحرف «س» .

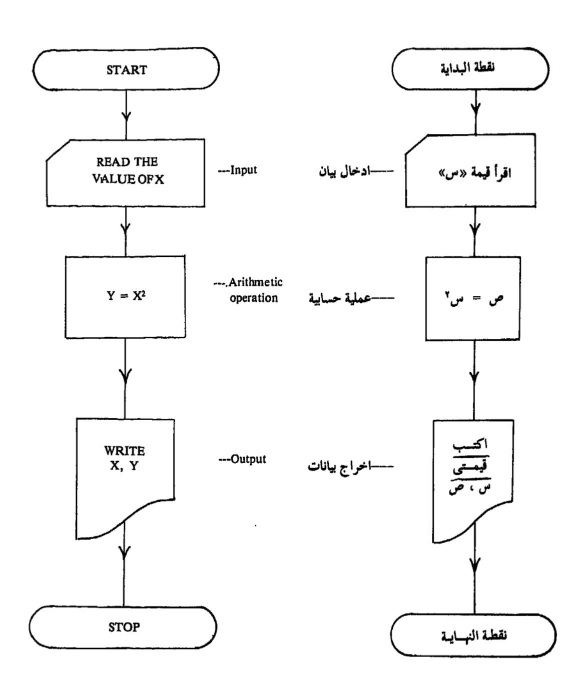
٢ – نوجد مربع «س» ولنرمز له بالحرف «ص» (عملية حسابية) .

٣ – نكتب قيمتى «س» و «ص» (اخراج بيانات) .

هذه الخطوات المـكتوبة البسيطة تساعدنا كثيراً في رسم مخطط التدفق للبرنامـج والموضح في الشكل (٢-٢) .

من الملاحظ أن المخطط السابق قد وضع باللغتين العربية والانجليزية لتوضيح وتيسير الأمور في البداية ، ولكننا من الآن فصاعداً سنضع جميع مخططات التدفق باللغة الانجليزية فقط لأن كتابة البرامـج ستكون بلغة الفورتران التي تكتب باللغة الانجليزية .

ولنحاول الآن أن نرسم مخطط التدفق لحل مشكلة أكثر صعوبة وتعقيداًمن المشكلة السابقة .



الشكل (٢-٢) مخطط التدفق لعملية تربيع رقم

مثال (٣) :

لنفترض أن البيان المعطى في المثال السابق أصبح يتكون من مائة رقم بدلاً من رقم واحد وطلب منا أن نرسم مخططا تدفقيا يوضح سير العمليات الحسابية والمنطقية لحل مشكلة كهذه باستعمال الحاسب .

الحـل

ان من أكثر الطرق العملية والمنطقية للتفكير في حل مشكلة كهذه هي طريقة الآلة الحاسبة التي يمكن تلخيصها فيما يلي :

١ – ادخل الرقم الأول من البيان في الآلة الحاسبة .
٢ – اضغط زر التربيع أو اضرب الرقم في نفسه .
٣ – سجل مربع الرقم الناتـج .
٤ – ادخل الرقم الثاني من البيان في الآلة الحاسبة .
٥ – كرر الخطوات الثانية والثالثة .
٦ – كرر الخطوات الثانية والثالثة لكل رقم من الأرقام المتبقية بعد ادخال كل رقم على حده حتى يتم استكمال البيان .

مما سبق يتضح لنا أن مقدرتنا الطبيعية على احصاء عدد الأرقام التي تم معالجتها من البيان في وقت ما سوف يمكننا من التوقف بعد الانتهاء من البيان تماماً ، أي ليس من المعقول أن نتوقف بعد خمس وتسعون رقما فقط . اذا فللانسان بطبيعته عدادا أو مؤشرا يساعده على التوقف عند الخطوة المناسبة وهذه الخاصية لايمتلكها الحاسب الآلي لكنه من الممكن لنا أن نكسبه هذه المقدرة على الاحصاء اذا مااستعملنا مؤشراً أو عدادا يمكنه من التوقف في الوقت المناسب ، وهذا ماسنضعه في الاعتبار عند رسم مخطط التدفق لهذا المثال .

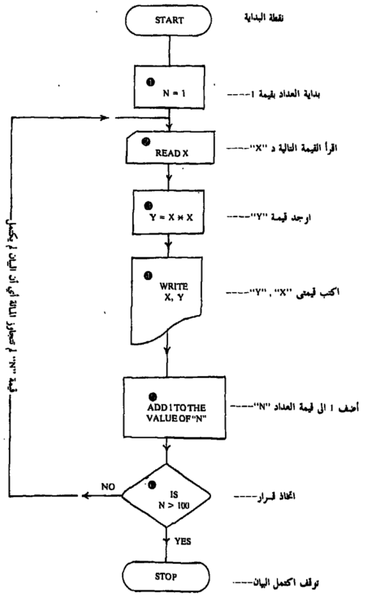
أما الآن فلنكتب الخطوات العملية التي نريد تنفيذها في الواقع لحل المشكلة والتي يمكن ترتيبها كالتالي :

١ – نبدأ باعطاء العداد ولنرمز له بالحرف «N» القيمة المبدئية 1 .
٢ – نقرأ أول رقم في البيان ولنرمز له بالحرف «X» .
٣ – توجد مربع قيمة «X» ولنرمز له بالحرف «Y» .
٤ – تكتب قيمتى «X» و «Y» .
٥ – أضف مقدار واحد الى قيمة «N» السابقة لتصبح القيمة الحالية مساوية القيمة الأولى مضافاً اليها 1 .

٦ - تجري العملية المنطقية التالية : اذا تجاوزت قيمة «N» الحالية المائة فأننا نأمرُ الحاسب بالتوقف لأنه استكمل المطلوب منه . أما اذا لم تتجاوز قيمة «N» المائة فأننا عندئذ نقوم بقراءة القيمة التالية في البيان والتي تمثل القيمة الجديدة للمتغير «X» .

٧ - تكرر الخطوات من ٣ الى ٦ .

بعد هذا العرض الموجز للخطوات المطلوبة نقوم برسم مخطط التدفق للعمليات الحسابية والمنطقية كما هو موضح في الشكل (٢-٣) .



الشكل (٢-٣) مخطط التدفق لبرنامج لايجاد مربعات مائة رقم

« تمـــريـن »

اكتب وصفاً مبسطاً بالأضافة الى مخطط التدفق لبعض العمليات التالية :

- اصلاح بنشر سيارة .
- شراء زوج من الأحذية .
- حساب معدلك التراكمي في الجامعة .
- الحصول على رخصة قيادة خصوصية .
- حساب دخلك السنوي .
- الذهاب في رحلة الى أبحر .

٢-٢ بعض العمليات الأساسية للحاسب الآلي :

بعد هذه المقدمة البسيطة عن مخطط التدفق للعمليات المنطقية والحسابية نصبح الآن في مرحلة نستطيع فيها أن نبدأ بكتابة برنامج بسيط بلغة الفورتران – ولكن قبل أن نبدأ ذلك نريد أن نتساءل عن العمليات والمهام الأساسية التي يستطيع أن يؤديها الحاسب الآلي عن طريق تعليمات وأوامر تشكل في مجموعها مايسمى بالبرنامج ؟ وفي الواقع فأن هذه العمليات يمكن تقسيمها الى ثلاث أنواع رئيسية هي :

أولاً : عمليات ادخال واخراج البيانات : وهذه احدى جوانب تفوق الحاسب الآلي فهو يقوم بمعالجة البيانات الواردة اليه مهما كانت ضخمة وطويلة . وتنتهي عادة هذه المعالجة ببيانات صادرة تمثل النتائج المطلوبة من البيانات الواردة بعد إجراء سلسلة من العمليات الحسابية والمنطقية عليها والتي تختلف من برنامج لآخر حسب الدور المطلوب منها . ففي المثال الأخير نلاحظ أن البيان الوارد يتكون من مائة رقم أما البيان الصادر فيتكون من مائتى رقم تشمل الأرقام الواردة ومربعاتها . وهاتين العمليتين تتمان عن طريق استعمال التعليمات .

« أقرأ » (بيان وارد) READ

« اكتب » « وأطبع » (بيان صادر) WRITE AND PRINT

ثانياً : العمليات الحسابية والمنطقية : وتشمل عمليات الحساب المعروفة من جمع وطرح وضرب وقسمة وأسس وغير ذلك من العمليات الرياضية كاللوغاريتمات والجذر التربيعي والدوال المثلثية ... الخ . أما العمليات المنطقية فهي تشمل عمليات المقارنة بين مقدارين أو أكثر ينتج على أثرها قرار معين يعتمد على نتيجة المقارنة فمثلاً قد تتم المقارنة بين مقدارين «س» و «ص» ومن ثم فأن قرارا

معيناً سوف يتخذ في حالة ما اذا كانت قيمة «س» أكبر من قيمة «ص» وقد يتخذ قرارا آخر مختلف تماماً في حالة ما اذا كانت قيمة «س» أصغر من قيمة «ص» وهكذا .

ثالثاً – عمليات التحكم : وهي تشمل العمليات التي تغير من المسار الطبيعي للبرنامج كما تشمل تعليمات التنفيذ المشروطة وتعليمه ايقاف البرنامج وغيرها . وسنتكلم فيما بعد عن هذه التعليمات بالتفصيل . فيما يلي سنحاول أن نستقرىء بعض العمليات المذكورة أعلاه من خلال البرنامج التالي :

٢-٣ برنامج حساب الاستحقاق السنوي :

مسألة :

اكتب برنامجاً يحسب الاستحقاق السنوي وكذلك صافي الاستحقاق السنوي لشخص يبلغ راتبه الشهري ٩٦٥٠ ريالاً سعودياً يدفع منها ٩٪ رسوم تقاعد كما يحصل على بدل مواصلات نقدي مقداره ٦٠٠ ريال سعودي .

تمرين

أرسم مخططاً تدفقياً يوضح مسار العمليات الحسابية والمنطقية للمسألة أعلاه !

الحل

الشكل (٢-٤) يمثل البرنامج المطلوب .

```
      REAL SALARY, TRANSP, TAX, GROSS, NET
      READ (5,*) SALARY, TRANSP, TAX
      GROSS = 12.Ø * SALARY + 12.Ø * TRANSP
      NET = GROSS - 12.Ø * TAX * SALARY
      WRITE (7,2Ø) GROSS, NET
2Ø    FORMAT (F 8.2, 5X, F7.2)
      STOP
      END
```

الشكل (٢-٤) البرنامج المطلوب لحل المسألة أعلاه

من الشكل السابق يبدو الأمر لأول وهلة وكأنه موضوع انشاء باللغة الانجليزية رغم أن المحتوى قد لا يعني الكثير بالنسبة لنا في هذه المرحلة ، ولكننا نستطيع أن نتلمس طريقنا خلال السطور وأن نفهم بعض ما يدور في هذا البرنامج البسيط رغم أننا لانملك القدرة حالياً على كتابة أي برنامج

فورتران . على أي حال سوف نحاول من خلال الصفحات القليلة القادمة أن نشرح كل جملة من جمل هذا البرنامج على حدة بصورة مبسطة تعطي انطباعا عاما عن شكل البرنامج المكتوب بلغة الفورتران .

فالسطر الأول يُعلِم الحاسب الآلي بجميع اسماء المتغيرات التي سوف تستعمل في البرنامج فمثلاً استعملت كلمة SALARY كاسم للمتغير الذي يمثل الراتب الشهري للشخص بينما استعملنا كلمة TRANSP (اختصار كلمة TRANSPORTATION) لترمز للمتغير الذي يمثل بدل المواصلات الشهري وكذلك الحال بالنسبة للمتغير GROSS الذي يعني إجمالي المستحق و TAX الذي يمثل ضريبة التقاعد ، أما المتغير NET فيمثل صافي الاستحقاق السنوي . وللملاحظة فأن هذه القائمة تسبقها كلمة REAL وهي توضح للحاسب أن مقادير هذه المتغيرات يجب أن تكون أرقاماً ذات فواصل عشرية ، وسوف نتكلم عن هذا الموضوع باسهاب في حينه .

أما السطر الثاني فهو عبارة تعليمة لأدخال بيان يشمل ثلاث مقادير لثلاث متغيرات هي SALARY ، TRANSP ، TAX ويتم هذا عن طريق كلمة READ كما أسلفنا سابقاً .

أما السطرين الثالث والرابع فهي عمليات حسابية بسيطة ينتج على أثرها حساب قيمتى المتغيرين GROSS و NET بمعرفة مقادير المتغيرات الأخرى والتي عُرفت عن طريق ادخال البيان في السطر الثاني . يلي ذلك تعليمة لأخراج بيان يشتمل على قيم المتغيرات المطلوب حسابها وهي كما نعرف من السؤال تمثل حلاً للسؤال المعطي ، والتعليمة تبدأ بكلمة WRITE والتي تعني كما أسلفنا « إكتب » ثم تكتب قائمة بأسماء المتغيرات المطلوب كتابة قيمها حسب ترتيبها في تعليمة الكتابة وحسب الطريقة الموضحة في الجملة التالية والتي تحمل الرقم 20 في بدايتها مع ملاحظة أن هذا الرقم موجود في تعليمة WRITE السابقة لجملة FORMAT والتي تعني في العربية « هيئة » أو « صفة » أو « طريقة » .

أما السطر التالي الذي يشتمل على كلمة STOP والتي تأمر الحاسب بأيقاف التنفيذ فوراً وعدم القيام بتنفيذ أي تعليمة من الآن فصاعدا ، وفي النهاية فأن كلمة END تعني النهاية الحسية للبرنامج حيث لا يمكن كتابة أي تعليمة بعد ذلك .

وفي الواقع فأنه يتضح لنا بعد مراجعة العمليات الأساسية التي يمكن أن يقوم بها الحاسب الآلي أن هذا البرنامج يعتبر نموذجا بسيطاً يشمل أمثلة مختلفة من هذه العمليات الأساسية .

أما الآن وقد أشبعنا فضول الطالب المهتم عن شكل البرنامج ومحتوياته والتي قد يتخيلها الطالب لأول وهلة أنها لابد وأن تكون جمل وكلمات سحرية عجيبة توجه الى الحاسب الآلي فيتصرف ازاؤها بتعقل وحكمة تفوق تعقل وحكمة أي كائن بشري ولكن الواقع أن الحاسب الآلي أغبى من

أي مخلوق في الوجود فهو عبارة عن آله صماء لاتملك حولاً ولاقوة ولا تجيد تفكيراً أو تدبيراً . ولذلك كان لابد من اعطاء هذه التعليمات بدقة بالغة فأي اختلال في هذا التعليمات حتى ولو كان خطأ املائيا فأنه قد يؤدي الى توقف البرنامج كليةً دون تنفيذ أي جزء منه وفي أحسن الظروف قد يؤدي هذا الخطأ الى الحصول على نتائج خاطئة وربما مضحكة وغير معقولة .

لذا فأننا ننصح الطالب الراغب في فهم لغة الفورتران أو أي لغة حاسب أخرى أن يكون دقيقاً حريصاً واعياً مدركاً لحقائق عمل البرنامج وقواعد اللغة التي كتب بها وكيفية تنفيذ الحاسب لخطواته حتى يستطيع تلافي كثير من الأخطاء الغير ضرورية والتي تضيع كثيراً من الجهد والوقت والمال .

٢-٤ طريقة كتابة برنامج ما :

أما كيفية تقديم البرنامج الى الحاسب الآلي ليقوم بمهمة تنفيذه واظهار النتائج المتوقعة منه فيتم عن طريق تقديم البرنامج بإستعمال الوحدة الطرفية Terminal وهي أكثر الطرق شيوعاً الآن لكونها عملية ومريحة واقتصادية في ذات الوقت . وقد تختلف طرق إستعمال الجهاز من شركة الى أخرى ولكنها كلها تدور حول نفس الفكرة الأساسية التي يمكن تلخيصها فيما يلي :

١ – تشغيل الجهاز SWITCH ON
٢ – الدخول في النظام LOG ON
٣ – كتابة البرنامج WRITING PROGRAM
٤ – تقديم البرنامج SUBMISSION
٥ – تنفيذ البرنامج EXECUTION

وجميع هذه الخطوات ماعدا الخطوة الثالثة يمكن اعتبارها أمور فنية تختلف من نظام الى آخر ولهذا فلن نخوض في تفاصيلها بل نتركها لأستاذ المادة ومساعده الفني والنظام المستعمل في الجامعة أو المعهد الذي يتبعه الطالب .

أما كتابة البرنامج فتكاد تكون متماثلة في جميع الأنظمة تقريباً ولهذا فسنتناولها هنا ببعض التفصيل لأهميتها . ففي لغة الفورتران يتكون السطر الواحد من ٨٠ خانة (أو ١٢٠ خانة في بعض الأجهزة) الا أن الخانات الفعلية المستعملة هي الخانات من ١ الى ٧٢ وهذه الخانات موزعة الى مناطق عمل رئيسية على الوجه التالي :

– الخانات من ١ الى ٥ تستعمل لتحديد رقم الجملة (إن وجدت الحاجة الى ذلك) ويسمى هذا المجال « مجال الترقيم » LABEL field ، ويمكن وضع أي رقم موجب صحيح في هذا المجال . ولا يمكن لجملتين مختلفتين أن تحملان نفس الرقم .

- الخانة رقم ٦ تترك عادة خالية BLANK الا في حالة واحدة فقط هي حالة كون المعلومات المكتوبة في الخانات ٧ الى ٧٢ تعتبر مواصلة للجملة التي تسبقها في السطر السابق . أي أن وجود أي رمز CHARACTER ماعدا الفراغ أو الصفر في الخانة ٦ يعني أن السطر الحالي ماهو الا إمتداد للسطر السابق لأن السطر السابق لايتسع للجملة بأكملها . ويسمى هذا السطر CONTINUATION LINE « السطر المتواصل » . ولايجوز شغر « مجال التعريف » في حالة شغر الخانة ٦ لأن رقم الجملة هو نفس رقم الجملة السابقة بإعتبارها اتصالا لها . وسوف نرى في أمثلة قادمة كيف أن بعض الجمل تشغر أكثر من ٦٤ خانة (٧٢-٦ = ٦٦) .
- الخانات من ٧ الى ٧٢ هي الخانات التي توضع فيها التعليمة أو الجملة طبقاً لقواعد لغة الفورتران ويسمى هذا المجال بمجال التعليمات «STATEMENT field» .
- يجوز استخدام الخانات من ٧٣ الى ٨٠ لأغراض أخرى غير ذات علاقة بمضمون الجملة المكتوبة في الخانات من ٧ الى ٧٢ . وأهم هذه الأغراض هي المعلومات التعريفية الأضافية للبرنامج .
- في حالة وضع الحرف C **في الخانة الأولى** ينتج لدينا مايسمى « بسطر الملاحظات «COMMENT Line» وهذا السطر يتجاهله الحاسب الآلي تماماً من الناحية التنفيذية ، لكنه يقوم بطباعته فقط عندما يكتب البرنامج . ولهذا فيمكن للمبرمج أن يستعمل هذه الميزة الهامة بشرح بعض خطوات البرنامج شرحاً إملائياً لغوياً دون الخضوع لقواعد لغة الفورتران المحدودة ، وذلك عن طريق كتابة حرف C في الخانة الأولى كما أسلفنا ثم كتابة أي مادة يرغبها المبرمج في الخانات من ٢ الى ٨٠ .

فمثلاً نستطيع أن نبدأ البرنامج المعطى في الشكل (٢-٤) بالسطور التالية :

الخانة الأولى

```
C  *** THIS PROGAM ISMADE BY A BEGINNER * $ *+*
C                      * * * *
C
C  . . .  THIS PROGRAM COMPUTES THE ANNUAL SALARY AND THE
C  . . . . . . .  NET ANNUAL SALARY OF A GOVERNMENT
C  . . . . . . . . . . .  EMPLOYE.
```

ويمكن تلخيص ماسبق في المستطيل التالي :

ينقسم السطر في لغة الفورتران الى أربع أقسام رئيسية يسمى كلاً منها مجالاً وهي كالتالي :

(أ) مجال الترقيم LABEL field (من الخانة ١ الى ٥)

(ب) مجال الاتصال CONTINUATION field (الخانة ٦)

(جـ) مجال التعليمات STATEMENT field (من الخانة ٧ الى ٧٢)

(د) مجال التعريف IDENTIFICATION field (من الخانة ٧٣ الى ٨٠)

Continuation field

Identification field

Label field

Statement field

1 567 72 73 80

٢-٥ تسمية المتغيرات :

مما سبق يتضح لنا أن المهمة الرئيسية للبرنامج هي معالجة البيانات تلقائياً وفق التعليمات المعطاه له وهذه الخاصية من أهم ما يميز الحاسب الآلي عن الآلة الحاسبة ، كما علمنا من الفصل السابق أن البيانات الواردة تخزن أولاً في الذاكرة ثم يقوم الحاسب بمعالجة هذه البيانات طبقاً للتعليمات الصادرة اليه عن طريق البرنامج لينتج لنا في النهاية البيانات الجديدة المطلوبة . ففي المثال السابق كان البيان الوارد يشتمل على قيم ثلاث متغيرات هي TAX, TRANSP, SALARY وهذه القيم تحفظ في الذاكرة تلقائياً حتى يتسنى لنا استعمالها فيما بعد في عمليات حسابية أخرى أو أن يُطلب كتابة بيان يشتمل على قيم هذه المتغيرات كما فعلنا في السطر الخامس من البرنامج .

اذاً الذاكرة جزء مهم من الحاسب الآلي ، لذا وجب علينا أن نضع تصوراً منطقياً وواقعياً عن طريقة تخزين البيانات في الذاكرة ومعالجتها وتخزين النتائج فيها لاستعمالها فيما بعد ، وأبسط هذه التصورات هي تصور الذاكرة كوحدة تشتمل على أعداد هائلة من الخلاياCELL المتماثلة تكون في بادىء الأمر بدون أسماء . وعند اطلاق اسم على متغير فأن الحاسب يقوم تلقائياً بتعيين خلية في الذاكرة تحمل ذلك الأسم باستمرار حتى انتهاء تنفيذ البرنامج . فمثلاً في المثال السابق كانت هناك خلية باسم SALARY وخلية باسم TRANSP وخلايا بأسماء NET, GROSS, TAX وهذه

الخلايا تحمل هذه الأسماء بصفة دائمة حتى انتهاء البرنامج وفي الواقع فإنه لايهمنا موقع الخلية في الذاكرة بقدر مايهمنا معرفة أن هناك خلية ما تحمل اسماً معيناً ثابتاً ، ولكن متى تكون الكلمة صالحة لأن تكون إسماً لخلية من خلايا الذاكرة ؟ فكما أن اختيار اسم الطفل المسلم يخضع لشروط معينة فكذلك الحال بالنسبة لتسمية الخلية ، أما هذه الشروط فهي كالتالي :

١ - أن يبدأ الاسم بحرف من حروف اللغة الانجليزية (A - Z) .
٢ - أن يتكون الاسم من حروف (A - Z) أو أرقام (Ø - 9) فقط .
٣ - لايجوز أن يتكون الاسم من أكثر من ستة رموز . ومقصد بالرموز مجموعة الحروف أو الخليط من الحروف والأرقام بشرط أن يتحقق الشرط ١ .

مثال (٤) :

الأسماء التالية تعتبر صحيحة وشرعية في لغة الفورتران :

A, X, TF, N3, ALI, Z2B4, JAMAL, USA6, EGYPT

مثال (٥) :

لايجوز استعمال الأسماء التالية في لغة الفورتران ، لماذا ؟

MAHMOOD, 4X, C4 BF, MAJID$, X+T, U.S.A.

٢-٦ قيمة المتغير :

لنفرض أن لدينا خلية تحمل الاسم SALARY ولنتصور أن الخلية لها شكل مستطيل (أنظر الشكل (٢-٥) أ) .



الشكل (٢-٥) أ

الشكل (٢-٥) ب

ولنفترض أن الحاسب أعطى قيمة SALARY عن طريق قراءة بيان يشتمل على قيمة SALARY ولنفرض أن هذه القيمة هي 9650.0 . فالذي يحدث هو أن هذه الخلية التي تحمل الاسم SALARY تحتوي في الواقع بداخلها على القيمة العددية 9650.0 (أنظر الشكل (٢-٥) ب) . اذاً يمكننا أن نشبه الخلية بكأس فارغ يحمل اسماً معيناً ثم نضع فيه ماءاً مثلاً ليكون لدينا كأساً له اسم معين

ويحتوي على سائل هو الماء . ولكن من الواضح أنه يمكننا إستبدال محتوى الكأس من الماء بسائل آخر وليكن عصير برتقال مثلاً وفي هذه المرحلة بالذات نرى أن اسم الكأس لم يتغير بينما تغير محتوى الكأس تماماً ، وكذلك الحال بالنسبة للخلية التي تحمل اسماً معيناً خلال تنفيذ البرنامج – فالأسم لايمكن تغييره بينما يمكن تغيير محتوى الخلية بأحدى التعليمات المناسبة .

مثال (٥) :

الشكل (٢-٦) يمثل التغيير الحاصل في محتوى خلية تحمل الاسم X .

X	X	X
– 4.00	10.00	

(شكل (٢-٦))

قاعدة هامة : خلال تنفيذ برنامج معين لايمكن تغيير إسم خلية ما بينما يمكن تغيير محتوى الخلية أي أنه لايمكن تغيير إسم المتغير بينما يمكن تغيير قيمته حسب تعليمات البرنامج .

ومن المهم أن ننوه هنا بأن الحاسب يقوم بحجز خلية لكل متغير يخالف في اسمه للمتغيرات التي سبق ظهورها في البرنامج . فعلى سبيل المثال فأن المتغيرات .

B12A, AB21, AB12,A21B, A12B, BA21, BA12, B21A

وان تشابهت في مجموعة الرموز المكونة لأسمائها الا أن كلا منها تختلف في اسمها عن الآخر مما يترتب عليه قيام الحاسب بحجز خلية لكل منها .

٢-٧ أنواع قيم المتغيرات : Data types

هناك نوعان مختلفان من القيم العددية التي يمكن أن تحتويها خلية تحمل اسم متغير ما ، وهما كالتالي :

أولاً : الأرقام الحقيقية (العشرية) : REAL numbers

وهي الأرقام التي تحتوي على نقطة عشرية واحدة وقد تكون موجبة أو سالبة . ويطلق عليها أيضاً اسم « الأرقام ذات العلامة العشرية المتحركة » Floating-point numbers .

ثانياً : الأرقام الصحيحة : INTEGER numbers

وهي لاتحوي أي اجزاء عشرية كما لاتحوي أي نقاط عشرية ، أما أن تكون سالبة أو موجبة .

مثال (٦) :

تعتبر الأرقام التالية أرقاماً حقيقية بالنسبة للغة الفورتران .

.0001   1146.9   −517.352   +61.20   −1.00   1.   .1

بينا لايجوز استعمال الأرقام التالية كأرقام حقيقية في لغة الفورتران .

1,146.9   78   −639 1/5   +1   −1/2

مثال (٧) :

تعتبر الأرقام التالية أرقاماً صحيحة في لغة الفورتران :

1469   −85   +034   −156   1

بينا لايجوز معاملة الأرقام التالية على أساس أنها أرقاماً صحيحة في لغة الفورتران :

1,469   −85.   +4/2   −47.0   1.

لاحظ أن 1 و .1 يختلفان تماماً في نوعيتهما فالأول رقم صحيح والثاني رقم حقيقي مع أنهما لايختلفان رياضياً .

٢–٨ الجملة المبينة (التوضيحية) :   Declaration Statement

أما السؤال المتوقع الآن فهو كيف يمكن تحديد نوعية قيمة المتغير ؟ واختصاراً سوف نتكلم عن نوع المتغير نفسه فنقول متغير حقيقي (عشري) REAL Variable أو متغير صحيحINTEGER Variable والجواب على هذا السؤال يكمن في استعمال ما يسمى بالجملة المبينة (التوضيحية) Declaration Statement التي تبين أو تحدد نوع المتغير بالنسبة للحاسب الآلي ويمكن تلخيص عمل هذه الجملة في المستطيل التالي :

الجملة المبينة	Declaration Statement
الشكل العـام	TYPE list

أما كلمة TYPE فهي أما أن تكون REAL أو INTEGER بينا List تعني قائمة بأسماء المتغيرات مفصولة بفواصل «,» وهذه القائمة تتكون من متغير واحد أو أكثر .

المعنى : هذه التعليمة توجه الحاسب الآلي الى اطلاق الأسماء الموجودة في List على خلايا تستعمل لتخزين بيانات عددية من نوع TYPE المذكور في أول الجملة .

مثال (٨) :

```
REAL A, B3, X8Z61, NUMBER, FOUR4
INTEGER ALPHA, COUNT, N, M1, FX24Y
INTEGER EGYPT, SP, TABLE, L3
REAL SAUDIA, R4, KARL, MTXF, LONDON
INTEGER SIX
```

فالسطر الأول يعتبر بمثابة تعليمة للحاسب الآلي لاطلاق الأسماء الخمسة الموجودة في القائمة على خلايا ذات محتوى عشري ، أي أن A يعتبر متغيرا ذو قيمة عشرية وكذلك الحال بالنسبة لـ FOUR 4 NUMBER, X8Z61, B3 أما السطر الأخير فيؤكد أن الحاسب سيعامل المتغير SIX على أساس أنه رقم صحيح . وماقيل عن السطر الأول والأخير يقال تماماً عن بقية الأسطر . ولكن علينا أن نضع في الأعتبار أن المتغير ذو القيمة العشرية يجب أن ينظر اليه على أساس أنه متغير حقيقي « عشري » طوال البرنامج وكذلك الحال بالنسبة للمتغيرات الصحية .

« تمـــاريـن »

١ - أي من التالي يمكن إعتباره إسماً شرعياً لخلية ذاكره ؟ علل في حالة النفي !

23K5
ILOVE7
ABDULLAH
I LOVE
K. A. U.
FORTRAN

٢ - أي من الجمل التالية يمكن إعتباره جملة « مبينة » في لغة الفورتران ؟ علل في حالة النفي !

INTEGER A
INTEGER A, BOB, X, IN
REAL K, JO, XY4
REAL ALPHA, 14.6
INTEGER VERYLONG, SHORT
REAL, A

INTERGER BOX

REAL INTGER

٣ – اكتب جملة (جمل) فورتران تبين أننا نريد أن نستعمل خليتى ذاكره باسم ALI و NADER لتخزين ارقام صحيحة (INTEGERS) !

٤ – فيما يلي ضع اشارة (√) بجانب الرقم الحقيقي وضع دائرة حول الرقم الصحيح ؟

41.7

Ø6

784.

– 49

+ 896.673

Ø

ملاحظات هامة :

١ – المقادير الثابتة تنقسم الى فئتين هما فئة الأرقام العشرية REALS وفئة الأرقام الصحيحة INTEGERS .

٢ – المقدار المتغير يرمز له باسم يخضع لشروط التسمية في لغة الفورتران كما أوضحنا سابقاً ، وهذا لا يعني بالضرورة تغيير قيمة المتغير خلال البرنامج ، فمن المحتمل أن يكون للمتغير قيمة ثابتة لاتتغير طوال البرنامج .

٣ – لايجوز وضع اشارتين رياضيتين متعاقبتين بدون فصلهما بواسطة قوس . فمثلاً :

لايجوز كتابة 2 – *3 أو 2 – **5.Ø

بينما يجب كتابتها على نحو (2 –)*3 أو (2 –)**5.Ø

٤ – في حالة استعمال أرقام عشرية وأرقام صحيحة في نفس المقدار الجبري فأن الناتج يكون رقماً عشرياً .

فمثلاً ناتج المقدار الجبري 3 + 4.**2

يساوي 19.Ø

وهذا ما يسمى بـ MIXED MODE OPERATION

٥ – لايجوز رفع رقم سالب الى أس عشري مثل 1.5**(3.Ø –) أو 2.Ø**(2 –) بينما يجوز رفع رقم موجب الى أس عشري مثل 2.3**(.Ø5) أو (2.Ø –)**(1.67) والسبب في ذلك أن طريقة ايجاد الحاسب لمقدار مثل 2.5)**(4.15) تتم طبقاً للخطوات التالية :

– ايجاد لوغاريتم الرقم العشري 4.15 أي أوجد Ln 4.15

– ايجاد حاصل ضرب الأس 2.5 في المقدار Ln 4.15 أو أوجد 2.5 Ln 4.15
– ايجاد حاصل قيمة رفع العدد الطبيعي e الى الناتـج السابق لنحصل على النتيجة المطلوبة .
أي أن

$$(4.15)**2.5 \quad = \quad e^{(2.5 \text{ Ln } 4.15)}$$

وحيث أنه لايمكن حساب لوغاريتم أي رقم سالب لذلك كان من غير الممكن تكليف الحاسب بايجاد قيمة رقم سالب مرفوع الى أس عشري .

٦ – حاول قدر المستطاع استعمال الأسس الصحيحة بدلاًمن الأسس العشرية ، وذلك لدقة استعمال الأسس الصحيحة لأنها تعني ضرب الرقم في نفسه عدد مرات يساوي قيمة الأس بينما الأس العشري يؤدي بالحاسب الى استعمال الطريقة المذكورة في ٥ . اذاً هناك فارق جوهري في طريقة ايجاد قيمة كل من :

(3.Ø)**2 و (3.Ø)**2.

بالرغم من أن الناتـج يساوي 9.Ø في كلا الحالتين ، ولكن الوضع سيختلف في حالة ايجاد أسس لبعض الأرقام العشرية المعقدة فمثلاً ، في حالة ايجاد قيم كل من :

(35.6324Ø32)**2 و (35.6324Ø32)**2.Ø

سيكون هناك فارق في الناتـج بالتأكيد لأن الأس الصحيح يعني ضرب المقدار في نفسه بينما الأس العشري يعني ايجاد لوغاريتم الى تقريب معين ومن ثم رفع العدد e الى ضعف قيمة اللوغاريتم مما يؤدي الى اعطاء قيمة تقريبية مختلفة عن الحالة الأولى .

٢-٩ موقع الجملة المبينة :

أما موقع الجملة المبينة فيجب أن يكون في بداية البرنامـج أي في السطر الأول اذا كانت لدينا جملة واحدة فقط وإذا كانت هناك أكثر من جملة فيجب أن تكتب في السطور التالية مباشرة ، وليس هناك مانع من كتابة أي عدد من الجمل المبينة حتى وإن كان بالامكان الاستغناء عن بعض هذه الجمل فمثلاً يمكننا استبدال هاتين الجملتين الصحيحتين .

```
REAL X
REAL Y
```

بجملة واحدة صحيحة هي :

```
REAL X,Y
```

والواقع أن موقع الجملة المبينة في بداية البرنامـج هو موقع طبيعي جداً حيث أن الجملة تحدد نوع المتغير الذي سوف يستعمل فيما بعد في البرنامـج .

١-١٠ التخصص التلقائي :

لنفترض أننا استعملنا متغيراً دون تحديد نوعه بواسطة جملة مبينة فما الذي يحدث ياترى ؟ في الواقع أنه في هذه الحالة فأن الحاسب يقوم تلقائياً بتحديد نوع المتغير حسب القاعدة التالية :

١ – اذا كان اسم المتغير يبدأ بأحدى الحروف من A الى H أو من O الى Z فأنه يعامل معاملة متغير حقيقي (عشري) REAL .

٢ – اذا كان اسم المتغير يبدأ بأحد الحروف من I الى N فأنه يعامل وكأنه متغير صحيح INTEGER .

اذاً الجملة المبينة (التوضيحية) غير لازمة بالضرورة ولكن من الأفضل بالتأكيد إستعمال الجمل المبينة (التوضيحية) وذلك للأسباب التالية :

١ – أن الجمل المبينة بحكم موقعها في بداية البرنامج تؤدي الى حصر جميع المتغيرات التي سوف تستعمل في البرنامج وهذا يؤدي بالتالي الى سهولة قراءة البرنامج من قبل الآخرين كما يعطي انطباعاً جيداً عن المبرمج من حيث التنظيم والبساطة .

٢ – استعمال الجمل المبينة يطعى المبرمج مرونة في استعمال الأسماء المناسبة والتي يمكن أن تصف المتغير الوصف المناسب الذي يعطي قارىء البرنامج صورة واضحة عن ماهية المتغير وعن الغرض المستعمل له .

مثال (٩) :

لنفترض أننا نريد أن نختار اسماً لمتغير يمثل عدد الأميال التي تقطعها سيارة في خلال ساعة واحدة فبالنسبة للمبرمج الفطن فأنه من البديهي أن يختار اسم MILES لهذا المتغير والذي يعطي صورة واضحة عن المتغير ونفترض أن عدد الأميال يمثل بأرقام عشرية من نوع REAL ، ففي هذه الحالة يجب علينا استعمال الجملة المبينة REAL MILES .

والتي ليس لنا غنى عنها اذا أردنا استعمال هذا الاسم ذو المحتوى العشري . أما اذا لم نرغب في استعمال جمالة مبينة فأننا لانستطيع استعمال اسم MILES لأن الحاسب في هذه الحالة يعامل هذا

المتغير تلقائياً على أساس أنه رقم صحيح رغم أنه في الواقع رقم عشري . وبالتالي فأن هذا يؤدي الى استعمال الاسم الغير مناسب للمتغير المطلوب مما يجعل البرنامج أكثر صعوبة وأقل تنظيماً .

٢-١١ جملة التعيين : ASSIGNMENT STATEMENT

تكلمنا فيما مضى عن تسمية متغير ما وعرفنا أن التسمية أنما تعني تسمية خلية بأسم معين ، اما نوعية محتواها فهو خاضع للجملة المبينة التي تحوي المتغير في قائمتها والا فأنه يخضع للتحديد التلقائي حسب القاعدة السالفة الذكر .

ولكننا لم نتحدث عن طريقة ادخال تلك القيمة في الخلية التي تحمل ذلك الاسم علماً أنه من الضروري جداً ادخال تلك البيانات في خلاياها الخاصة حتى يتمكن الحاسب من معالجتها عن طريق وحدة العمليات الحسابية والمنطقية التي تتصل بهذه الخلايا الموجودة في الذاكرة ومن ثم تُجرى عليها العمليات الحسابية والمنطقية المطلوبة في البرنامج .

وفي الواقع فأن هناك طريقتان مختلفتان لايصال القيم العددية الى خلاياها الخاصة بها وأحدى هاتين الطريقتين مباشرة والأخرى غير مباشرة ولنبدأ بالطريقة المباشرة البسيطة : وتتلخص هذه الطريقة في ايصال قيمة المتغير الى خليته عن طريق تعيين تلك القيمة بواسطة معادلة حسابية تسمى « جملة التعيين » Assignment Statement وتعتبر هذه الطريقة مباشرة لأنها توضع ضمن سطور البرنامج في الشكل العام التالي :

Variable	=	Expression
↕		↕
المتغيـــر	=	مقدار جبري أو عملية حسابية .

أما المتغير فقد تكلمنا عن تسميته سابقاً . بقى لنا أن نتكلم عن المقدار الجبري واشارة المساواة « = » . أما المقدار الجبري فهو المقدار الذي يكتب على يمين اشارة المساواه وقد يأتي في أحدى الصور التالية :

١ – مقدار ثابت – ومثال ذلك X = 1.0

٢ – متغير واحد فقط ، ومثال ذلك X = Y

٣ – مقدار جبري يشمل بعض المتغيرات والثوابت التي يفصل بينها اشارات العمليات الحسابية المختلفة من أسس وجمع وطرح وضرب وقسمة وبعض العمليات الأخرى ومثال ذلك

X = (2. + Y) / Z

أما العمليات الجبرية الأساسية فهي كالتالي :

العملية الحسابية	الاشارة الرياضية	الاشارة في لغة الفورتران	مثال رياضي	المثال بلغة الفورتران
الجمع	+	+	$a + b$	A + B
الطرح	–	–	$x - y$	X – Y
الضرب	×	*	$a \times b$	A * B
القسمة	÷و–و/	/	$a/b, \frac{a}{b}, a \div b$	A/B
الأس		**	$(x)^y$	X ** Y

٢–١٢ ترتيب العمليات الحسابية : Order of Arithmetic Operations

ياترى كيف تتم عملية ترتيب العمليات الحسابية بواسطة الحاسب الآلي وما هو النظام الذي يتبعه في هذا الشأن . لنأخذ مثلاً المقدار التالي :

5 * 3 + 2

لو كان ترتيب العمليات يتم من اليسار الى اليمين فأن عملية الضرب تتم أولاً لنحصل على (15 = 5*3) ثم نضيف اليه 2 ليكون الناتج 17 . أما لو كان ترتيب العمليات معاكسا لهذا الأفتراض ، أي أن ترتيب العمليات يتم من اليمين الى اليسار لحصلنا على (5 = 2 + 3) ثم ضربنا الناتج في 5 لنحصل على الناتج 25 . اذا من البديهي أن نستنتج أن أحد الجوابين غير صحيح وبالتالي فأن ترتيب العمليات الحسابية لابد أن يكون مهماً جداً بالنسبة لدارسي الفورتران — وفي الواقع أن الترتيب يتم كالتالي :

أولاً : الأقواس ( )

ثانياً : الأس **

ثالثاً : الضرب والقسمة / و *

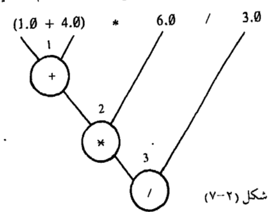
رابعاً : الجمع والطرح – و +

أما اذا تساوت عمليتين في الترتيب كعمليتى ضرب وقسمة أو عمليتى ضرب فأن التنفيذ يكون من الشمال الى اليمين .

مثال (١٠) :

أوجد ناتـج المقدار الجبري (1.0 + 4.0) * 6.0 / 3.0

الحــل : حسب قواعد الترتيب التي ذكرناها نستنتج أن الترتيب يتم كالتالي :



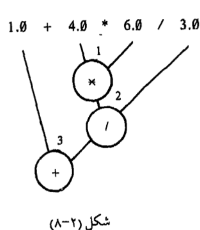
شكل (٢-٧)

وبالتالي فالناتـج هو 3.0 / 6.0 * 5.0 والذي يساوي 10.0 .

مثال (١١) :

أوجد قيمة المقدار الجبري 1.0 + 4.0 * 6.0 / 3.0 لاحظ أن هذا المقدار مشابه للمقدار السابق مع غياب القوس فقط .

الحــل :



شكل (٢-٨)

من الواضح أن الناتج سيكون حسب الشكل أعلاه مساوياً لـ

$8.0 + 1.0 = (24.0 / 3.0) + 1.0$

$9.0 =$

مثال (١٢) :

أوجد قيمة المقدار الجبري $4.0 * (3.0 * (X + (Y - Z)) / (A + B)) - 5.0 * (A - B) ** 2$

ولنفترض أن لدينا القيم التالية :

X = 1.0

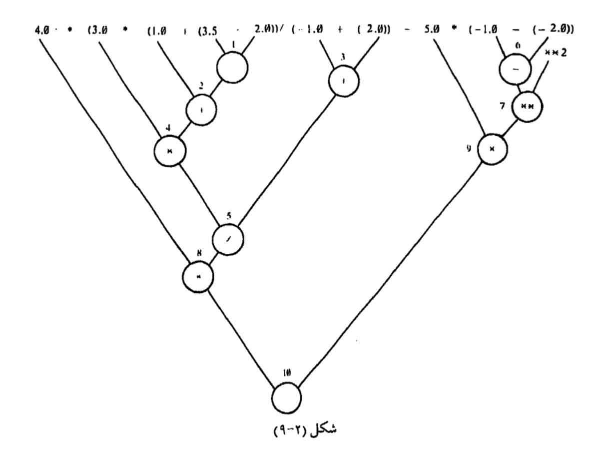
Y = 3.5

Z = 2.0

A = − 1.0

B = − 2.0

الحـــل :



شكل (٢-٩)

في الشكل أعلاه نستنتج أن المقدار سوف يصبح حسب ترتيب العمليات أعلاه :

4.0 * ((3.0 * 2.5) / (− 3.0)) − 5.0 * (1.0) * * 2

والذي يساوي :

4.0 * (− 2.5) − 5.0 * 1.0 = − 10.0 − 5.0 = − 15.0

ومن الملاحظ أنه اذا كانت هناك أقواس بداخل قوس فأن الحاسب يبدأ بتنفيذ العملية الحسابية في القوس الأصغر ثم الأكبر فالأكبر الى حد سبعة أقواس بداخل بعضها البعض بالنسبة لمعظم الحواسب الآلية . ومن الملاحظ أيضاً أن عدد الأقواس المفتوحة على الجهة اليمنى يجب أن يكون مساوياً لعدد الأقواس المفتوحة على الجهة اليسرى .

٢-١٣ ضرورة استعمال الأقواس :

لنفترض أنه طلب منا كتابة المقدار الرياضي التالي : $\frac{a + b}{c + d}$

ولنفرض أننا كتبنا المقدار في الشكل التالي A + B / C + D دون أن نهتم بوضع أي أقواس للبسط والمقام ، ولكن الواقع أن هذا المقدار يساوي A + (B/C) + D وهو مختلف تماماً عن المطلوب والذي يجب أن يكتب في الشكل (A+B) / (C+D) حتى يعطى المطلوب تماماً .

اذاً من الضروري الاستعانة بالأقواس لكتابة المقدار بالطريقة الصحيحة كما أن استعمال الأقواس وان لم يكن ضرورياً في بعض الأحيان الا أنه يساعد على توضيح المقدار الجبري وبخاصة اذا كان المقدار طويلاً ومعقداً .

مثال (١٣) :

المقدار A + B / C * D يساوي تماماً المقدار A + ((B / C)*D)

ولكن قراءة المقدار الثاني أسهل وأوضح ولا تدع مجالاً للتخمين اطلاقاً بالرغم من أن الأمر لايحتاج الى وضع أقواس على الأطلاق ولكن المبرمج الجيد يختار دائماً الطريق الواضح السهل على الآخرين سلوكه والاستفادة منه .

٢-١٤ اشارة المساواة :

بقي لنا بالنسبة لجمل التعيين أن نتكلم عن اشارة المساواة وما تعنيه بالنسبة للحاسب الآلي . ففي الرياضيات اذا كتبنا المعادلة التالية A = A + 1 ثم حاولنا حل هذه المعادلة فأننا نحصل على الحل اللامعقول ٠ = ١ .

ولكن الأمر يختلف بالنسبة للحاسب الآلي والسبب يكمن في تفسير معنى اشارة المساواة بالنسبة للحاسب فهي لا تعني المساواة تماماً كما يعرفه الجميع في علم الحساب وإنما تعني التالي :

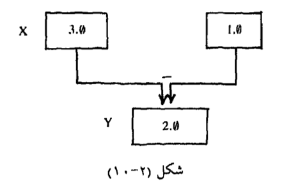
أوجد قيمة المقدار الجبري الموجود على يمين اشارة المساواة ثم ضع هذه القيمة الناتجة في الخلية التابعة للمتغير المكتوب على يسار اشارة المساواة . اما اذا كانت هناك قيمة سابقة للمتغير فأن الحاسب يقوم باستبدال القيمة السابقة بالقيمة الحالية الناتجة .

مثال (١٤) :

أفرض أن $Y = X - 1.0$ وأن قيمة x تساوي 3.0 ، أوجد قيمة Y ؟

الحـــل :

لنتصور أن كلاً من X و Y يمكن تمثيله بخلية على شكل مستطيل كما في الشكل (٢–١٠) وكما يظهر في الشكل أن قيمة Y تساوي 2.0



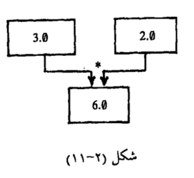
شكل (٢–١٠)

مثال (١٥) :

أفرض أن قيمة X الحالية تساوي 3.0 أوجد قيمة X من المعادلة الحسابية :

$$X = X * 2.0$$

الحـــل : من الشكل (٢–١١) يتضح أن قيمة X الجديدة تساوي 6.0 وأن هذه القيمة تحل محل القيمة القديمة لـ X والتي تساوي 3.0 والتي لايمكن استرجاعها حيث أنها فقدت تماماً بعد عملية استبدالها بالقيمة الجديدة .



شكل (٢-١١)

أنظر الى السطور التالية والتي تمثل جزءاً من برنامـج :

```
REAL X, Y, Z
     X  =  10.0
     Z  =  X + Y
     Y  =  5.0
```

في البرنامـج الموضح أعلاه ، عندما يصل الحاسب الى السطر الثالث فإنه سيقوم باستعمال أي قيمة سابقة موجودة في الخلية Y لحساب قيمة Z على الرغم من أن ذلك ليس هو الوارد في ذهن المبرمـج . ويلاحظ أيضاً أن الحاسب لن يعترض على تنفيذ هذا السطر .

```
REAL X, Y, Z
     X  =  10.0
     Y  =  5.0
     Z  =  X + Y
```

أي أنه عند تنفيذ أي عملية حسابية يجب أن تكون قيم جميع المتغيرات المستعملة في الطرف الأيمن من اشارة المساواة معلومة حتى يمكن ايجاد قيمة المتغير المكتوب في الطرف الأيسر من المعادلة .

« تمـــارين عــامة »

١ - في نهاية الجزء التالي من برنامج ما ، ماهي قيم كل من المتغيرين A و B ؟

```
INTEGER A , B
      B  =  10
      A  =  B
      B  =  2
```

٢ - ماهي القيم التي ستعطى للمتغير الصحيح B في كل من جمل التعيين التالية :

```
      B  =  2*3*4 / 4
      B  =  (2/1) + 1
      B  =  -18*3
      B  =  -4 * 5 + 2
```

٣ - أي من التالي يُعتبر جملة تعيين شرعية في لغة الفورتران ؟ علل في حال النفي ! (افترض أن جميع المتغيرات تمثل أرقاماً صحيحة » .

```
      A = A*A + A
      BAKR = 3
      MILE + FEET = 45
      CAT * DOG = - 16 + CAT
      HAMID = HASAN + FAHD
      LAI + JAMAL - 5
      3BOB = 476
      ALI = 3(15) + AHMED
      RICE = 16 + RICE
      OLIVE = FRTRAN / 4986132
      NO + YES = ORDER
```

٤ - ماهي القيم التي ستعطي للمتغير الصحيح FLOOS في كل من جمل التعيين التالية ؟

```
      FLOOS  =  2 + (8*3) / 4
```

```
FLOOS  =  323 / 2
FLOOS  =  (27 / 28) + 3
FLOOS  =  (21 / 20) - 1
FLOOS  =  (8/16) * 1000
```

٥ – اكتب جملاً بلغة الفورتران تقوم بالمهام التالية :

(أ) تبين أن أحد خلايا الذاكرة سيطلق عليها اسم ZAID وستعطى القيمة الصحيحة 2 .

(ب) تعين لـ ZAID قيمته السابقة مضروبة في نفسها ٥ مرات .

(جـ) تعين لـ ZAID قيمته السابقة بالأضافة الى Z .

٦ – أي من الأرقام التالية يعتبر « ثابت صحيح » ؟ علل في حالة النفي !

```
1                    3 * 4
1.Ø                  36.7
-13                  -129.4
-.987654321          376
6Ø  +  Ø             -Ø
```

٧ – أي من الأرقام التالية يعتبر « ثابت حقيقي » ؟ علل في حالة النفي ؟

```
1                    5.86
2.Ø                  5.68 + 3
-3.ØØ                -.ØØØØ1
-4.4 + 4.Ø           367
-3.Ø * 2             -15. + 4.Ø
```

٨ – اكتب جملة « فورتران » مساوية لكل جملة من الجمل الرياضية التالية :

1. $X + y^3$
2. $(X + y)^3$
3. X^4
4. $a + (b \div c)$
5. $\frac{a + b}{c}$
6. $a - \frac{bc}{a + d}$
7. $2x - 6y$
8. x/y^2
9. $(x+y)^2 / x^2 - y^5$
10. $\frac{ay}{3d} - bx$

٩ – أوجد قيمة المقادير التالية :

```
-1.0**4
3.0*3.**2
4/5 * 2 + 3.0
-3**2**3
(6.0 - 3.) / 4.0**2
2.*3.0 - 7./14.0+6.2
(2.0)**2.5
```

١٠– لنفرض أن A = – 1.0 علل لماذا تكون A**2 صحيحة بينما لاتعتبر A**2.0 شرعية في لغة الفورتران ؟

١١– استعمل الأقواس لتوضيح المقادير الجبرية التالية :

```
ALI + HAMID * FAHD ** 2
METER + CM - FEET * TIME + KM ** POWER ** DIST
L/N/M
```

١٢– في البرنامج التالي تتغير قيمة المتغير RESULT عدة مرات أثناء سير البرنامج . أوجد جميع هذه القيم ؟

```
INTEGER A,B,C
REAL RESULT , R1 , R2
R1 = 1
R2 = 2
A = 1
B = 4
C = 16
RESULT = R1 * R2
RESULT = A * R2
RESULT = B/(A*B)
RESULT = B - C
RESULT = - C/B
STOP
END
```

١٣– عند كتابة مقدار جبري يمكننا أن نستعمل الرمز ab لنعني حاصل ضرب a و b بينما لا يجوز ذلك في لغة الفورتران . علل ذلك ؟

الفصـــل الثالث

تعليمات إدخال وإخراج البيانات

الفصــل الثالث

تعليمات إدخال وإخراج البيانات Input / Output Statements

مـقـدمـة : ذكرنا فيما مضى أن جملة التعيين التي تأخذ الشكل العام .

Variable = expression

المقدار الجبري = المتغــير

تعطى للمتغير القيمة الناتجة من حساب قيمة المقدار الجبري ، وحيث أنه من الممكن أن يكون المقدار الجبري مقداراً ثابتاً constant فأنه من الواضح أن جملة التعيين التي تأخذ الشكل العام .

Variable = Constant

ثابت = المتغير

يمكن إعتبارها وسيلة مباشرة من وسائل إدخال البيانات الى الحاسب الآلي ، فمثلاً لو نظرنا الى الجزء التالي من برنامــج ما :

```
REAL X, Y, SUM
X = 2.0
Y = -1.0
SUM = X + Y
```

شكل (٣-١)

فانه من الواضح أن قيمتى X و Y قد زودت للحاسب عن طريق جملتى التعيين في السطرين الثاني والثالث ، وبالتالي فأن الحاسب يستطيع أن يزودنا بقيمة SUM والتي تساوي مجموع قيمتى X و Y كما هو موضح في السطر الرابع من البرنامــج ، وهكذا لابد لنا أن نعطي بعض قيم المتغيرات لكي نحصل على قيم جديدة نحتاجها .

فالعملية اذن عبارة عن أخذ وعطاء ولهذا فأن عملية إدخال البيانات يعتبر جزءاً أساسياً وعنصراً هاماً في مجال البرمجة الألكترونية كما هو واضح في المثال السابق .

لنفترض الآن أن بين إيدينا بيانا يشتمل على ألف قيمة لابد لنا أن نغذيها للحاسب الآلي حتى نحصل على بعض النتائـج المطلوبة كحاصل جمعها أو ضربها أو المعدل أو ماشابه ذلك . من البديهي أن يفكر القارىء المبتدىء في استعمال ألف جملة تعيين مختلفة ليزود الحاسب بالقيم الموجودة في

البيان ، ولكن من الواضح أن هذه الفكرة غير منطقية وغير عملية على الاطلاق لأنها في الحقيقة تفقد الحاسب الآلي أهم خصائصه على الاطلاق ألا وهي عملية اختصار الوقت والجهد ، لهذا كان ولا بد من وجود وسائل أخرى أكثر سرعة وبساطة وسهولة لادخال البيانات الى الحاسب بدون اضاعة الوقت والجهد في كتابة جمل تعيين بعدد القيم الموجودة في البيان .

٣-١ جملة القراءة The READ Statement

وتعتبر هذه الجملة من أهم الوسائل الغير مباشرة لادخال البيانات الى الحاسب الآلي لأن الجملة نفسها تكون ضمن نطاق البرنامج الفعلي أما البيان فيمكن التفكير فيه كملحق للبرنامج لايكون ضمن حدود البرنامج الفعلية .

البيان المرفق		البرنامج الفعلي
89.5 30 20.0 50 76.5 66 : 45 −20.5 30		: READ Statement : END

شكل (٣-٢)

أما الشكل العام لجملة اقرأ فهو كالتالي :

READ (n, m) V1, V2, V3, V4, ...

حيث n هو رقم الجهاز القارىء وعادة ما يكون 2 أو 5 .

m هو رقم الجملة الشارحة : FORMAT statement number

V1, V2, V3, ... قائمة بأسماء المتغيرات المطلوب قراءة قيمها من البيان Input Data حسب تسلسلها في القائمة .

ولنوضح قليلاً كل عنصر من عناصر جملة اقرأ . فالرقم n عادة ما يكون رقماً ثابتاً صحيحاً تحدده نوعية الجهاز المستعمل للقراءة (انظر ١-٢) ، فهناك قارىء للبطاقات «Card Reader» وهناك قارىء للاشرطة الممغنطة «Tape drive» وفي بعض الأحيان تكون القراءة عن طريق الالات الكاتبة عن بعد «Teletypes» أو عن طريق الشاشات والنوع الأخير هو النوع السائد الاستعمال حاليا

خصوصا للمبتدئين . أما الرقم m فهو رقم لجملة شارحة تشرح الطريقة التي كتبت بها القيم في البيان المرفق . فمن الجائز أن نكتب أربع قيم في السطر الأول وقيمة واحدة فقط في السطر الثاني وقد يختلف الوضع فتكتب قيمتين في السطر الأول وثلاث قيم في السطر الثاني ... وهكذا بالنسبة لبقية الأسطر في البيان ، لذا فان وحود الجملة الشارحة ضروري وهي جزء من البرنامج وسنتكلم عنها بالتفصيل فيما بعد .

اما قائمة المتغيرات فهي أما أن تتكون من متغير واحد ، أو أكثر من متغير مفصولة بفواصل «Comma» أما وظيفة جملة اقرأ فهي أنها تأمر الحاسب بقراءة القيم الموجودة في البيان المرفق طبقاً لوصف القراءة الذي تشترطه الجملة الشارحة ثم وضع هذه القيم في خلايا تحمل الأسماء الموجودة في القائمة التي تعتبر جزءا من الجملة نفسها . أما كيفية كتابة هذه القيم وترتيبها في البيان المرفق فهذا يعتمد تماماً على الجملة الشارحة التي تحمل الرقم m .

مثال (١) :

(أ) READ (5 , 2Ø) A, B,MAN

(ب) READ (5 , 3ØØ) METER, MILE

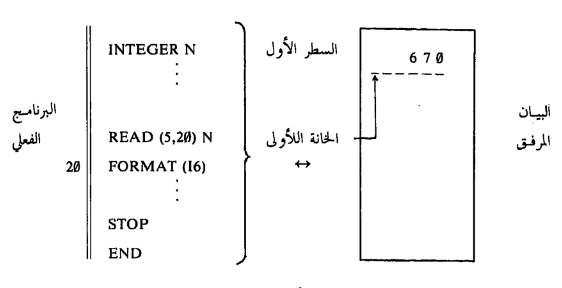
(جـ) READ (5 , 6441) X

٣-٢ وصف البيانات :

لكل نوع من أنواع القيم ، يوجد نوع خاص من واصفة البيانات «data descriptor» فالتعريف الذي يصف الأعداد الصحيحة «INTEGERS» يختلف عن التعريف الذي يصف الأعداد الحقيقية «REALS» فمثلاً الوصف «Iw» يُستعمل لوصف الأعداد الصحيحة أي أن I6 يستعمل للدلالة على عدد صحيح في البيان يشغل ست خانات في السطر المقروء ، أما كيف تتم هذه الدلالة فالأمر يكون واضحاً اذا مانظرنا الى المثال التالي :

```
     READ (5 , 2Ø) N
2Ø   FORMAT (I6)
```

فالجملة الأولى تأمر الحاسب بأن يقرأ قيمة N من البيان المرفق حسب الجملة الشارحة ذات الرقم 2Ø وعندئذ يقوم الجهاز القارىء للبيانات بقراءة قيمة N من خلال الست خانات الأولى في السطر الأول من البيان . أما الآن فلنمثل هذه العملية بيانياً حتى تتضح الصورة في ذهن القارىء :



الشكل (٣-٣)

ومما يلاحظ أن قيمة N المساوية لـ 670 لاتشغل سوى ثلاث خانات رغم أننا قد حجزنا لها عن طريق الجملة الشارحة ستة خانات ، ولكن هذا لا خلاف فيه طالما أن قيمة N قد كتبت في الوضع الصحيح . ونقصد بالموضع الصحيح ملأ الخانات التي في أقصى اليمين أولاً بأول ، فمثلاً لو كتبنا قيمة N على الشكل التالي :

_ 6 7 0 _ _

الخانة الأولى .

لاعتبرها الحاسب كما لو أنها 67000 أي أنه سوف يملأ الخانات التي في أقصى اليمين بأصفار . اذا لابد لنا أن نراعي هذه النقطة وأن نكون حذرين جداً عند كتابة قيم المتغيرات في البيان المرفق كما لابد لنا نراعي أن العدد السالب يجب أن تسبقه اشارة «-» والتي تأخذ إحدى الخانات المحجوزة لقيمة المتغير .

أما في حالة المتغيرات الحقيقية فإن واصف البيان يكون على الشكل Fw. d حيث w يساوي مجموع الخانات المحجوزة للعدد الحقيقي بينما d تساوي مجموع الخانات المحجوزة للجزء العشري من الرقم كما تحجز خانة واحدة للنقطة العشرية . وبالتالي فأن عدد الخانات المحجوزة للجزء الصحيح من الرقم تساوي w-d-1 .

مثـال :

اكتب الرقم الحقيقي 31.45- حسب واصفة البيان F7.2 .

الحــل :

مجموع الخانات المحجوزة للرقم هي ٧ خانات ، اثنتين منها للجزء العشري وواحدة للنقطة العشرية ، وبالتالي تتبقى ٧-٢-١=٤ خانات للجزء الصحيح الذي يتضمن الاشارة السالبة (أنظر الشكل (٣-٤)) .

```
 - 3 1 . 4 5
 - - - - - -
  w-d-1    d
```

الشكل (٣-٤)

اذا مما سبق نستنتج أن Iw تكون للأعداد الصحيحة بينما Fw.d تكون للأعداد الحقيقية .

مثــال : لنفترض أن لدينا الجزء التالي من برنامـج ما .

البرنامـج الفعـلي

```
    REAL XSUM
    INTEGER NUMBER
    READ (5,1) XSUM, NUMBER
1   FORMAT (F 5.2,I3)
       :
    STOP
    END
```

البيان المرفـق

```
4 Ø . 3 2 - 1 5
- - - - - - - -
```

الشكل (٣-٥)

لاحظ أن قيمة XSUM شغلت جيمع الخانات المخصصة لها وهي ٥ خانات وكذلك الحال بالنسبة لقيمة NUMBER ولكن ماذا سيحدث لو أن قيمة XSUM كانت فعلاً 4Ø.32- فأننا عندئذ يجب علينا أن نغير من واصف البيان F5.2 الى F6.2 لأن الواصف السابق لايكفي للقيمة التي تحتاج الى ستة خانات على الأقل . اذاً لابد لنا من مراعاة حجم القيم المستخدمة في البيان وبالتالي نستخدم واصف البيان الذي من الممكن أن يغطي جميع القيم الموجودة في البيان ، كأن تستعمل في هذا المثال F1Ø.2 أو أن نستخدم I6 بدلاً من I3 وهكذا مع مراعاة التقيد بالأنظمة المذكورة آنفاً من حيث كتابة البيان طبقاً لنصوص واصفات البيان في الجملة الشارحة .

٣-٣ جملة الطباعة : The WRITE Statement

تعلمنا حتى الآن كيف نطلق الأسماء على خلايا الذاكرة وتعلمنا كيف نعطي هذه الخلايا القيم الخاصة بها عن طريق جملة التعيين أو عن طريق جملة اقرأ ، وتعلمنا كيف نوجه الأوامر الى الحاسب ليؤدي بعض المهام الحسابية والجبرية – ولكننا لم نتعلم حتى الآن كيف نأمر الحاسب بطباعة النتائج المطلوبة ليتسنى لنا الاستفادة منها لأنه من الجائز أن يكون الحاسب قد قام بايجاد قيمة متغير ما وحفظ تلك القيمة في الخلية الخاصة بالمتغير ، ولكن حتى يمكننا الاستفادة من تلك النتيجة – يجب علينا أن نحصل على تلك القيمة مكتوبة أمامنا ، والا فأن ايجاد تلك القيمة أو عدمه يتعبر متساويان في نظر المبرمـج ولكانت كتابة البرامـج مضيعة للوقت والجهد والمال .

مثال : انظر الى البرنامـج التالي :

```
REAL X, Y, RESULT
X = 1.56
Y = -29.54
RESULT = Y**2 + X*Y
STOP
END
```

الشكل (٣-٦)

اذا مانظرنا الى هذا البرنامـج لأول وهله ، فأننا قد نعتقد أن البرنامـج قام بايجاد قيمة المتغير RESULT وأننا قد حصلنا على المطلوب . ولكن الواقع أن الحاسب يكون قد أوجد فعلاً – بعد تنفيذ البرنامـج – قيمة المتغير RESULT ومن المستحيل بالنسبة لنا أن نطلع على خلايا الذاكرة لنحصل على هذه القيمة وعندئذ ندرك أن تنفيذ هذا البرنامـج وعدم تنفيذه سيان بالنسبة لنا .

لذلك كان من البديهي أن تكون هناك تعليمة توجه الحاسب الى كتابة أو طباعة القيم والنتائج التي يرغب المبرمـج في الحصول عليها بعد تنفيذ برنامـج معين . أما الوسيلة لتحقيق ذلك فهو استعمال جملة اكتب ولكي ندرك وظيفة هذه التعليمة بوضوح ونفهم ماهيتها فهما شاملا ، يتوجب علينا أن ندرك الأجزاء المختلفة من هذه التعليمة .

في البداية علينا أن ندرك (كما هو الحال بالنسبة لجملة اقرأ) أن هناك أجهزة مختلفة يمكن ايصالها بالحاسب لتقوم بمهمة الكتابة (انظر ١-٢) ، ومن هذه الأجهزة الطابعات الخطية Line Printers والأشرطة الممغنطة «Magnetic Tapes» وغير ذلك من الأجهزة الا أن الجهاز الشائع الاستعمال بالنسبة للطالب المبتدىء هو الطابعات الخطية أو الوحدات الطرفية «Teletypes» .

ولكي يصبح بالأمكان للحاسب أن ينفذ تعليمة اكتب فأنه وكما هو الحال بالنسبة لتعليمة اقرأ يجب علينا أن نحدد نوع الجهاز المستعمل للكتابة كما يجب علينا أن نحدد الجملة الشارحة FORMAT Statement وعلينا أن نوضح أيضاً أسماء المتغيرات المطلوبة كتابة قيمها . لذا فأن الشكل العام لجملة « اكتب » هو التالي :

WRITE (n , m) V1, V2, V3, ...

حيث n هو رقم الجهاز المستعمل للكتابة وغالباً ما يكون «6»

m هو رقم الجملة الشارحة FORMAT Statement number

V1, V2, V3, ... قائمة باسماء المتغيرات المطلوب كتابة قيمها حسب تسلسلها في القائمة .

أما المعنى العام فهو كالتالي : استعمل الجهاز ذو الرقم n لكتابة قيم المتغيرات ... V1, V2, V3 بالتسلسل وحسب الوصف المرفق في الجملة الشارحة ذات الرقم m .

أمثـلـة :

```
WRITE (6,1Ø36) A,B,CX
WRITE (6,713) SUM
WRITE (6,2Ø4) X1, X2
WRITE (6,593)
```

أما الآن فنتكلم قليلاً عن الجملة الشارحة FORMAT Statement بقليل من التفصيل ، ووظيفة الجملة الشارحة هي شرح طريقة كتابة النتائج المطلوبة سطراً سطرا ، وهناك عدة طرق لتحديد موقع السطر التالي من البيانات الناتجة ، أما أهمها فهي الطرق التالية :

أولاً : أن يكتب السطر التالي بعد السطر الحالي مباشرة .

ثانياً : أن يترك سطر خال بعد السطر الحالي ثم يكتب السطر التالي .

ثالثاً : أن يكتب السطر التالي في بداية الصفحة التالية مع ترك باقي الصفحة الحالية خاليا .

ويتم تحديد أي من هذه الاختيارات باستخدام ما يسمى برمز التحكم الحركي «Carriage control character» والتي تكتب ضمن الجملة الشارحة بين قوسين صغيرين وكما هو موضح في الشكل (٣-٧) .

' '	(فراغ واحد) : يعني تحرك الى السطر التالي في نفس الصفحة قبل أن تبدأ الطباعة .
' 0 '	(صفر) : يعني اترك السطر التالي خالياً ثم أبدأ الطباعة في السطر الذي يليه .
' 1 '	(واحـد) : يعني إترك بقية الصفحة الحالية خاليةً ثم ابدأ الطباعة في السطر الأول من الصفحة التالية .

الشكل (٣-٧)

أما موقع هذا الرمز فيجب أن يكون في بداية الجملة الشارحة لكي تشعر الجهاز الطابع بموقع السطر التالي طبقاً لرغبة المبرمـج .

مثـال :

```
    WRITE (6,251) X, Y
251 FORMAT ('0', F4.1, F5.2)
```

من النظرة الأولى ندرك أن المطلوب هو طباعة قيمتي Y و X حسب الجملة الشارحة ذات الرقم 251 . وبمجرد أن يجد الجهاز الطابع الرمز '0' فأنه يدرك أن عليه أن يترك سطرا خالياً ثم يبدأ الكتابة في السطر الذي يليه حيث يقوم بطباعة قيمتي X و Y حسب واصفات البيان الموضحة .

والآن لنفترض أننا نرغب من الحاسب أن يكتب جملة كتابة حرفية دون تغيير أو تبديل ، وذلك بقصد توضيح وتسهيل قراءة نتائج البرنامـج «Program Output» أو لأي غرض آخر ، فعندئذ كل ما نحتاجه هو وضع الجملة المطلوب طباعتها حرفياً بين أقواس صغيرة Quote marks كما فعلنا سابقاً بالنسبة لرموز التحكم الحركي (انظر الشكل ٣-٧) .

مثـال : انظر الى البرنامـج التالي :

```
    X = 456.14
    Y = -310.52
    SUM = X + Y
    WRITE (6, 100) X, Y, SUM
100 FORMAT ('  ', 'THE SUM OF', F7.2, 'AND', F8.2,'IS', F7.2)
    STOP
    END
```

أما الناتـج الذي سوف نحصل عليه فسيكون كالتالي :

الخانـة الأولى

↑ THE SUM OF ƀ 456.14 ƀ AND ƀ −310.52 ƀ IS ƀ 145.62

حيث ƀ تعني blank أو خانة فارغة . وبهذا يتضح لنا أن استعمال الواصفات الحرفية «Literal descriptors» تجعل قراءة الناتـج اكثر وضوحاً للقارىء الذي يطلع على نتائـج البرنامـج للأستفادة منها رغم أن المبرمـج لايجد صعوبة في قراءة الناتـج وذلك لأنه خبير بالبرنامـج مطلع على كل صغيرة وكبيرة فيه ، ولكن كما أسلفنا أن المبرمـج يكتب البرنامج لِيستفيد هو منه شخصياً كما يستفيد منه الآخرون بنفس المقدار لهذا فلو أن البرنامـج السابق كتب بنفس الطريقة السابقة مع استبدال الجملة الشارحة رقم 100 بالجملة :

100 FORMAT (F7.2, F8.2, F7.2)

لكان الناتـج على الشكل التالي :

الخانـة الأولى

↑ 456.14 ƀ −31.52 ƀ 145.62

والذي تصعب قراءته على من يقرأ الناتـج لأول مرة رغم بساطة البرنامـج ورغم أن النتيجة هي نفس النتيجة السابقة فما بالك عندما يكون البرنامـج متشابكاً ويحتوي على عشرات الأسطر من النتائـج المختلفة التي تحتاج الى ايضاح وتبسيط .

اذا الواصفات الحرفية Literal descriptors لاتغير شيئاً من النتائـج ولكنها تؤدي دوراً كبيراً في سبيل تبسيط قراءة نتائـج البرنامـج حتى على المبرمـج نفسه ، وخاصة عندما يعود الى قراءة البرنامـج بعد زمن طويل من كتابته واستعماله .

فيما سبق استعرضنا بعض الجمل الشارحة وكيفية استخدامها في إدخال البيانات أو إستخراج النتائـج ، ولكن من المعروف أن لغة الفورتران تحوي الكثير من الامكانيات التي يمكن استخدامها في تنظيم إدخال البيانات أو كتابة النتائـج بالصورة التي يراد الحصول عليها . وفيما يلي سنستعرض بإستخدام أمثلة مختلفة ، بعض الجمل الشارحة الأخرى التي تستخدم في لغة الفورتران والتي منها :

٣-٤ إستخدام الحرف A في الجملة الشارحة :

يستخدم الحرف A في الجملة الشارحة اذا ما أريد قراءة أو طباعة بيانات تحتوي على حروف أو أرقام أو كليهما معاً ، ولذا يرمز الحرف A الى كلمة Alphameric . وتتم عملية تخزين الحروف والأرقام بإستخدام الحرف في عدد صحيح من الخلايا ، أي تتم عملية التخزين بتقسيم البيان الى عدد

متساوٍ من الحروف والأرقام ، وكل مجموعة تخزن في خلية واحدة . ومعظم الحاسبات لاتقبل أكثر من أربعة أرقام وحروف في كل خلية ، ومع ذلك اذا كانت مجموعة الحروف والأرقام التي يراد تخزينها في الخلية الواحدة أكبر من سعة تلك الخلية ، فأنها لن تقبل سوى مجموعة الحروف والأرقام التي في أقصى اليمين . أما اذا حدث العكس وكانت عدد الحروف والأرقام أقل من سعة الخلية فأن مجموعة الحروف والأرقام تلك تخزن في أقصى شمال الخلية ويتم استكمال باقي الخلية بأصفار . فعلى سبيل المثال إذا كانت كل خلية تتسع لأربعة حروف وأرقام فأنه إذا كان :

البيان المراد قراءته (تخزينه)	واصف البيان	البيان المخزون
HODA	A4	HODA
SARWAT	A6	RWAT
OK	A2	OKbb

وعند إستخراج بيانات سبق تخزينها ، بأفتراض أيضا أن كل خلية ستحتوي على عدد من الحروف والأرقام لايزيد عن أربعة ، فأنه اذا كان :

البيان المراد إستخراجه (طباعته)	واصف البيان	البيان المستخرج (المطبوع)
KING	A4	KING
KING	A2	KIbb
KING	A6	bbKING

مثـال ١ :

اذا أريد إدخال (قراءة) البيان FACULTY OF SCIENCE فإنه يتم تقسيم ذلك البيان الى مجموعات من الحروف ، كل مجموعة لاتزيد عن أربعة حروف وبالتالي تكون تعليمة القراءة كالتالي :

```
      READ (5,9) NA1, NA2, NA3, NA4, NA5
9     FORMAT (5A4)
```

نلاحظ في هذا المثال أننا قسمنا البيان الى خمسة أقسام كل منها لايزيد عن أربعة حروف . أي أن البيان سيشغل على الأقل خمس خلايا مختلفة ، ونظراً لأن الخلايا مختلفة لذا أعطينا إسماً لكل منها وهي :

NA5 , ... , NA2 , NA1

مثـال ٢ :

سبق تخزين البيان :

KINGbABDbAZIZbUNIV.

في المتغيرات V1 , V2 , ... , V25 بحيث أن كل حرف كان قد سبق تخزينه في خلية واحدة . اكتب جملة الطباعة التي يمكنها تحقيق ذلك :

```
      WRITE (6,4) V1, V2, ... , V25
4     FORMAT (1X, 25A1)
```

٣-٥ إستخدام الحرف E في الجملة الشارحة :

عادة مايستخدم الحرف E في الجملة الشارحة ، عندما تكون النتائج المراد إستخراجها صغيرة جداً أو كبيرة جداً . فعلى سبيل المثال اذا كانت قيمة المتغير A التي تم حسابها وتخزينها في الحاسب هي Ø.ØØØØØ8 . فأذا أردنا طباعة قيمة المتغير A بالصيغة F6.4 مثلاً فأن النتيجة ستكون كالتالي :

(Ø.ØØØØ) . كذلك إذا كانت قيمة A هي 5763.426 وأردنا طباعة تلك القيمة بنفس الصيغة السابقة ، فإن النتيجة ستكون كالتالي(*) :

****.* وهذا يعني أن عدد الأرقام في الجزء الصحيح من النتيجة اكثر من العدد المراد إستخراجه بصيغة الطباعة ، وعلاج مثل تلك الحالات بسيط إذا كانت قيم المتغيرات في الحدود المعقولة والتي يمكن قراءتها كأن نكتفي بطباعة ثمانية أرقام عشرية بجانب القيمة الصحيحة للمتغير والتي قد تكون عشرة أرقام أخرى أو أقل . وفي مثل تلك الأحوال يمكننا تجنب الأخطاء التي قد تحدث نتيجة إستخدام صيغة طباعة غير كافية بإستخدام صيغة طباعة كبيرة نسبياً مثل F2Ø.8 مثلاً . ولكن في بعض المشاكل قد تكون قيم المتغيرات التي يتم حسابها داخل الحاسب كبيرة جداً مثل حسابات المشاكل الفلكية ، وقد تكون تلك القيم صغيرة جداً مثلما يحدث في المشاكل الفيزيائية أو الكيميائية والتي قد تأخذ قيم المتغيرات فيها القيمة $١٠^{-٢٠}$ أو أقل من ذلك . وفي جميع مثل تلك الحالات (إذا كان هناك تخوف من أن النتيجة قد تكون صغيرة أو كبيرة أو يصعب تقديرها) ، فأننا نستخدم صيغة الأس Exponent لطباعة تلك القيم والتي يرمز لها بالرمز E في الجملة الشارحة . فالقيم التالية يمكن كتابتها بصيغة الأس كالتالي :

(*) بعض الحاسبات تعطى خطأ في مثل تلك الحالات .

0.000008 $= 8.x10^{-6}$ = 8.0E-06 = 8.0E-6
$= 0.8x10^{-5}$ = 0.8E-05 = 0.8E-5
$= 0.08x10^{-4}$ = 0.08E-04 = 0.08E-4
= ---
$= 0.00008x10^{-1}$ = 0.00008E-01 = 0.00008E-1
$= 0.000008x10^{0}$ = 0.000008E00

كذلك فأن العدد :

9876543.21 $= 9.87654321x10^{6}$ = 9.87654321E06
$= 98.7654321x10^{5}$ = 98.7654321E05
$= 987.654321x10^{4}$ = 987.654321E04
= --
$= 9876543.21x10^{0}$ = 9876543.21E00
$= 98765432.1x10^{-1}$ = 98765432.1E−01
$= 987654321.0x10^{-2}$ = 987654321.0E-02

– من المثالين السابقين يمكن القول بأنه لقراءة أي عدد حقيقي في صورة أسية فأن العدد يتكون من جزئين يفصل بينهما الحرف E .

– الجزء الأول مجموعة من الأرقام (حسب درجة التقريب المراد الحصول عليها) وتحوي فيما بينها **العلامة العشرية** ومجموعة هذه الأرقام قد تكون موجبة أو سالبة وتسبق الحرف E .

– الجزء الثاني ويسمى بالأس Exponent ويكون عامة في الصورة E ± XX وفي حالة ما إذا كان الأس موجباً فأن الطابع يهمل كتابة الاشارة ، بينما يكتب الاشارة (−) في حالة ما إذا كان الأس سالباً . وتختلف القيمة العظمى للأس مابين حاسب وآخر ، ففي الحاسبات IBM 1130 أو UNIVAC 1108 فأن قيمة الأس لن تزيد عن 38 بينما في الحاسبات IBM 360/370 لاتزيد عن 75 .

ولاعطاء أي قيمة من القيم السابقة الى الحاسب لقراءتها ، فأن صيغة القراءة بإستخدام الحرف E تشبه الى حد كبير الصيغة بإستخدام الحرف F .

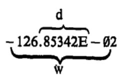
مثـال : اذا كانت :

A = −1.2685342 , B = 225.346

فأن القيم السابقة يمكن كتابتها باستخدام صيغة الأس E كالتالي :

المتغير	القيمة	القيمة بصورة F	القيمة في صورة أسية	القيمة باستخدام E
A	−1.2685342	F1Ø.7	−1.2685342E+ØØ	E14.7
			−12.685342E−Ø1	E14.6
			−126.85342E−Ø2	E14.5
			⋮	⋮
			−Ø.12685342E+Ø1	E15.8
			−Ø.Ø12685342E+Ø2	E16.9
			⋮	⋮
B	225.346	F 7.3	225.346E+ØØ	E11.3
			22.5346E+Ø1	E11.4
			⋮	⋮
			2253.46E−Ø1	E11.2
			22534.6E−Ø2	E11.1
			225346.ØE−Ø3	E12.1
			ØØØ225346.ØE−Ø3	E15.1

ويتضح من هذا المثال أنه لقراءة قيمة المتغير بالصورة E فأنه يقرأ في الصورة العامة Ew. d حيث w تمثل عدد جميع الأرقام التي يحتوي عليها العدد بما فيها حرف E واشارتي العدد والأس والعلامة العشرية ، أما d فتمثل عدد الأرقام التي تتلو العلامة العشرية . فمثلاً



أما عند طباعة قيمة المتغير المخزونة داخل الحاسب ، فأن هناك اختلافاً بسيطاً يحدث وهو أن الحاسب يقوم بتخزين العدد وطباعته بحيث أن العلامة العشرية تسبق أول رقم معنوي في العدد (أي عدد يختلف عن الصفر) ، بينما تتحدد قيمة العدد بقيمة الأس . فمثلاً العدد −126.85432E−Ø2 يمكن طباعته بالحاسب بصيغ مختلفة منها :

صيغة الكتابة	صورة العدد بالصيغة المطلوبة
E14.8	−.12685342E+Ø1
E12.3	ƀƀƀ−.127E+Ø1
E1Ø.4	−.1269E+Ø1
E 9.2	ƀ−.13E+Ø1
E2Ø.12	ƀƀ−.126853420000EØ1
E 7.2	*******

في صيغة الكتابة الأخيرة لن تطبع قيمة المتغير نظراً لأن صيغة الكتابة لهذا المتغير تحتوي على :-

- الأس واشارته وحرف E (٤ وحدات)
- رقمين عشريين (٢ وحدة)
- العلامة العشرية (١ وحدة)

وتبقى إشارة العدد نفسه التي لم يعمل حساب لها نظراً لأن صيغة الكتابة غير كافية . ولذا عندما يراد طباعة قيمة أي متغير يجب الأخذ في الأعتبار : تخصيص ٤ وحدات للأس وإشارته وحرف E بالأضافة الى تخصيص وحدة للعلامة العشرية ، وكذلك وحدة لاشارة العدد . وهذه الوحدات تعتبر أساسية عند كتابة قيمة أي متغير باستخدام الصيغة E ويضاف الى تلك الوحدات عدد الأرقام المطلوبة طباعتها ويتوقف ذلك على درجة التقريب المطلوب إستخراج العدد به .

٣-٦ إستخدام الحرف H في الجملة الشارحة :

يستخدم الحرف H(*) في الجملة الشارحة عندما يراد كتابة أية عناوين تحتوي على حروف أو أرقام أو أية علامات خاصة . ويشترط لاستخدامها معرفة **عـدد** الحروف والأرقام والعلامات الخاصة المراد كتابتها وكتابة ذلك العدد قبل الحرف H .

مثـال ١ :

```
      WRITE (6,15) R1,  R2
15    FORMAT (23H ƀ THE ƀ REAL ƀ ROOTS ƀ ARE ƀ X1 = , F8.2, 5X,
     1 7HAND ƀ X2 = , F8. 2)
```

عند تنفيذ تلك التعليمة سنجد أن النتيجة ستكون في الصورة :

أول عمود في الطابع

ƀTHEƀREALƀROOTSƀAREƀX1 = xxxxx.xxƀƀƀƀƀANDƀX2 = xxxxx.xx

* يرمز الحرف H الى العالم الاحصائي هوليرث Hollerith إعترافاً بفضله في إبتكار نظام البطاقات المثقوبة كوسيلة للتخزين وكان ذلك في عام ١٨٨٩م .

نلاحظ من المثال السابق أن :

١ – عدد الحروف والأرقام والعلامات الخاصة المراد كتابتها أولا هي ٢٣ ، ثم يكتب قيمة R1 وبعد ذلك يراد ترك خمس مسافات فارغة (5X) ثم يكتب عدة حروف وأرقام وعلامات خاصة أخرى عددها ٧ يتلوها مباشرة كتابة قيمة R2 .

٢ – إن القيم العددية التي يراد كتابتها هي قيم R1 ، R2ولو أنها في الجملة الشارحة أخذت أسماء X1 ، X2 وهذا لن يغير من الأمر شيئاً إذ أننا في الجملة الشارحة يمكن أن نكتب مانريده ولكن العبرة بأسماء المتغيرات الموجودة في تعليمة الكتابة ، ولذا لابد وأن تكون قيم R1 ، R2 هي المعروفة في البرنامـج وليس قيم X1 ، X2 .

مثـال ٢ :

```
   ||  WRITE (6,3)
3  ||  FORMAT (1X,10(1H*), 11HTHEbRESULTS,5(2H - +))
```

فعند تنفيذ تلك التعليمة فستكون نتيجة الطباعة كالتالي :

أول عمود في الطابع

```
↓
b**********THEbRESULTS.- + - + - + - + - +
```

يلاحظ في هذا المثال مايلي :

١ – اننا استخدمنا الأقواس في الجملة الشارحة للتعبير عن التكرار ، فقبل كتابة الجملة THE RESULTS يراد كتابة العلامة * عشر مرات ، وبعد كتابة هذه الجملة يراد كتابة العلامتين – ، + خمس مرات متتالية .

٢ – الجملة الشارحة السابقة تكافىء تماماً الجملة :

```
3  ||  FORMAT (1X,31H**********THEbRESULTS - + - + - + - + - +)
```

نلاحظ من المثالين السابقيين أن إستخدام الحرف H في الجملة الشارحة يماثل تماماً إستخدام الأقواس الصغيرة quote marks الا أن الأخير أكثر تطوراً وإستخداماً حيث إستخدامها لايتطلب معرفة عدد الحروف والأرقام والعلامات الخاصة ولكن نكتب ببساطة مايراد كتابته بين تلك الأقواس الصغيرة . ولكن تتضح أهمية إستخدام الحرف H عندما يراد كتابة مجموعة من الحروف أو الأرقام أو العلامات الخاصة عدداً معيناً من المرات . وعلى سبيل المثال اذا أردنا من الحاسب أن يطبع خطاً مكوناً من العلامة (–) عددها سبعون ، يمكننا تنفيذ ذلك ببساطة بكتابة :

```
   | WRITE (6,15)
15 | FORMAT (1X,70 (1H -))
```

أو

```
15 | FORMAT (1X,35 (2H - -))
```

أو

```
15 | FORMAT (1X,14 (5H - - - - -))
```

وهكذا .

٣-٧ إستخدام الحرف X في الجملة الشارحة :

يستخدم الحرف X في الجملة الشارحة عندما يراد من الحاسب ترك مسافات عند القراءة أو الكتابة . فعلى سبيل المثال ، لنفترض الجزء التالي من برنامج مـا :

```
   | PI = 3.14
   | R = 5. 268
   | AREA = PI*R*R
   | WRITE (6,18) R,PI,AREA
18 | FORMAT (1X,'R = ', F5.3,3X, 'PI = ', F4.2,5X, 'AREA = ', F7.4)
```

فعند تنفيذ الحاسب لهذا الجزء من البرنامج سنجد أن تنفيذ تعليمة الكتابة ستكون كالتالي :

أول عمود في الطابع
↓

```
bR =  5.268bbbPI = 3.14bbbbb ARE A = 87.1407
```

مثـال ٢ :

```
   | A = -2.157
   | B = 13.24
   | S = A + B
   | D = B / A
   | WRITE (6,14) A,B,S
   | WRITE (6,18)A,B,D
14 | FORMAT (1X,3(F6.3,2X))
18 | FORMAT (1X,3(2X,F6.3))
```

عند تنفيذ الحاسب لهذا الجزء من البرنامج ، سنجد أن النتائج ستكون كالتالي :

أول عمود في الطابع
↓

```
b-2.157bb13.240bb11.083bb
bbb-2.157bb13.240bbb-6.138
```

يلاحظ في هذا المثال أن هناك جملتين شارحتين أحداهما لوصف طريقة كتابة قيم المتغيرات A،B،S والأخرى للمتغيرات A،B،D . وفي كلتا الجملتين نلاحظ مايلي :

١ – بدء عملية الطباعة بترك مسافة واحدة فارغة (1X) قبل كتابة قيم المتغيرات ، وينصح بذلك عند كتابة أي بيانات أو نتائج باستخدام الطابع .

٢ – في الجملة الشارحة التي رقمها 14 نطلب فيها من الحاسب بأن يكتب قيم المتغيرات A،B،S بحيث أن قيمة كل متغير تكتب في الصورة F6.3 ثم **تتبعهـا** مسافتين فارغتين ، بينما في الجملة الشارحة التي رقمها 18 فاننا نطلب فيها من الحاسب بأن يكتب قيم المتغيرات A،B،D بحيث أن قيمة كل متغير تكتب في الصورة F6.3 وتكون **مسبوقة** بمسافتين فارغتين .

٣ – اننا إستخدمنا الأقواس للتعبير عن أن طريقة الطباعة الموجودة بين القوسين ستتكرر عدداً من المرات يساوي 3 .

مثـال ٣ :

```
        READ (3,5) A,B,C,D
5       FORMAT (F6.2,2X, F4.1,2(3X,F8.3))
```

فلتنفيذ تعليمة القراءة تلك ، يجب اعطاء قيم المتغيرات A،B،C،D للحاسب كما يلي :

```
xxx.xxb bxx.xb bbxxxx.xxx bbbxxxx.xxx
قيمة A  قيمة B  قيمة C      قيمة D
```

٣–٨ استخدام علامة القسمة / (Slash) في الجملة الشارحة :

وجود العلامة / في الجملة الشارحة يعني ببساطة الانتقال الى السطر التالي سواء لقراءة بيان عند تواجدها في الجملة الشارحة الخاصة بتعليمة قراءة أو لكتابة بيان عند تواجدها في الجملة الشارحة الخاصة بتعليمة كتابة .

مثـال ١ :

```
     READ (5,16) A,B,C,D
16   FORMAT (F8.2,2X,F6.3/F5.1, 3X, F9.3)
     WRITE (6,18) A,B,C,D
18   FORMAT (1X,'A = ', F8.2, 'B = ', F6.3, 3X, ' C = ',F5.1/34X,
   1 'D = ',F9.3/44 (1H = ))
```

عند تنفيذ تعليمة القراءة لهذا الجزء من البرنامـج ، فأن ذلك يتطلب اعطاء قيم المتغيرات D'C'B'A الى الحاسب لقراءتها بحيث تكون قيمتى B'A على سطر واحد وفي الصورة :

xxxxx.xxȸȸxx.xxx

A قيمـة B قيمـة

وقيمتى D,C في السطر الذي يليه ، وفي الصورة :

xxx.xȸȸȸxxxxx.xxx

C قيمـة D قيمـة

كما أنه عند تنفيذ تعليمة الكتابة ، فان الحاسب سيطبع قيم المتغيرات كالتالي :

أول عمـود
↓

ȸA = xxxxx.xxȸȸȸȸȸ B = xx.xxxȸȸȸ C = xxx.x D = xxxxx.xxx
==

مثـال ٢ :

```
     WRITE (6,2) A, B
2    FORMAT (1X,F7.2/////1X,F9.3)
```

في هذا المثال يراد من الحاسب كتابة قيمة المتغير A في سطر في الصورة xxxx.xx وبعد ذلك يترك **أربعـة** اسطر ثم يكتب قيمة المتغير B في الصورة xxxxx.xxx .

وفي المثاليين السابقين نلاحظ عدم وجود فاصلة (و) قبل أو بعد العلامة / .

الفصـــل الـرابـــع

تعليمـــــات التحـــكم

الفصــل الرابــع

تعليمـات التحـكم : «Control Statements»

مقدمـــة :

علمنا فيما سبق أن البرنامج المكتوب بلغة ما هو عبارة عن مجموعة من التعليمات التعاقبية والمتتالية والتي تهدف في النهاية الى ايجاد القيمة العددية لعملية حسابية قد تكون بسيطة أو معقدة ، وقد يتم حسابها في تعليمة واحدة أو بعد تجزئتها الى مجموعة تعليمات مستقلة ثم يتم تجميعها لتعطي في النهاية قيمة العملية الحسابية . فعلى سبيل المثال :

مثـال ١ :

اذا أردنا حساب قيمة المتغير Y والذي يتم حسابه عن طريق المعادلة التالية :

$$Y = \frac{-b + \sqrt{b^2 - 4aC}}{2a}$$

فأن ذلك يمكن أن يتم مثلاً عن طريق :

١ – كتابة المعادلة السابقة بلغة الفورتران :

```
Y = (-B+(B*B-4.*A*C)**Ø.5) / (2.*A)
```

٢ – تجزئة مكونات المعادلة واعادة تجميعها بعد حساب تلك المكونات كلا على حدة :

```
R = B*B-4.*A*C
A2=2.*A
S = - B+R**Ø.5
Y = S/A2
```

والطريقتان السابقتين ستعطيان نتيجة واحدة لقيمة المتغير Y ، اذا كانت قيم المتغيرات C,B,A مناسبة . أي اذا كانت قيمة A لاتساوي الصفر وكانت قيمة (B*B-4.*A*C) غير سالبة . فاذا

كانت قيمة A2 تساوي صفرا فسنجد أن قيمة Y نظريا في الطريقة الأولى تساوي مالا نهاية ، كما أن قيمة A2 في الطريقة الثانية ستساوي صفرا وقيمة Y (نظريا) ستساوي مالا نهاية . وكما نعلم فأن القيمة (مالا نهاية) هي قيمة غير معلومة يختلف تقديرها من شخص الى آخر ، ولكن هل سيستطيع الحاسب اعطاؤنا نتيجة محددة لقيمة المتغير Y في الطريقتين السابقتين ؟ والاجابة على ذلك سيكون بالنفي ، اذ أن الحاسب أيضا لن يتمكن من تقدير القيمة مهما كان الحاسب كبيرا .

كذلك سنصل الى نفس الاجابة من الحاسب اذا كانت قيمة (B*B – 4.*A*C) سالبة ، حيث أن الحاسب لن يعطينا نتيجة محددة لجذر كمية سالبة . ومثل هذه الحالات يقوم الحاسب باكتشافها في مرحلة التنفيذ للبرنامج «Execution» والتي كما سبق أن عرفنا أنها المرحلة التي تلي ترجمة البرنامج واكتشاف الأخطاء اللغوية به (Compilation) . ولذا فأن معظم الحاسبات تعتبر ظهور تلك الحالات كنوع آخر من الأخطاء Errors يقوم الحاسب عند اكتشافها في البرنامج بالتوقف عن تنفيذ التعليمات التي تليها وتحذير واضع البرنامج من نوع الخطأ الذي وقع فيه .

ولعدم الوقوع في مثل تلك الأخطاء عند كتابة برنامج وتنفيذه وجب على المبرمج أن يعطي الحاسب التعليمات المناسبة التي تجعله يتحاشى الوقوع في مثل تلك الأخطاء وذلك بأن يعطي الحاسب التعليمات المناسبة التي تجعله :

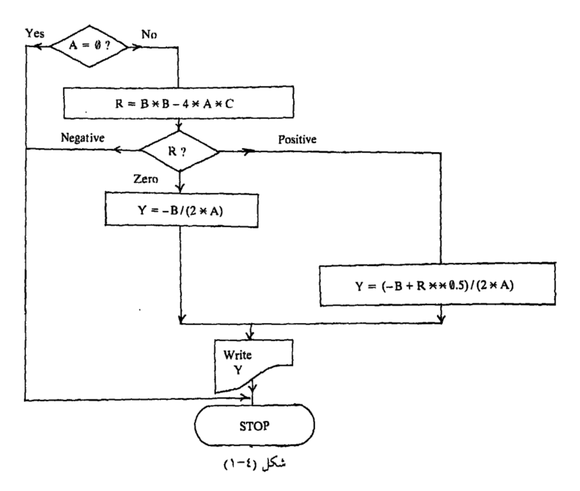
١ – يقوم باختبار المتغير أو الكمية التي يمكن أن تكون سبباً في الوقوع في أحد أنواع تلك الأخطاء .

٢ – يتوجه الى التعليمة أو مجموعة التعليمات المناسبة في البرنامج ، دون أن نجعل الحاسب يقوم بتنفيذ التعليمة التي تكون سبباً في ظهور مثل هذا النوع من الأخطاء .

ففي المثال السابق ، اذا كان المبرمج خذراً وماهراً كان عليه أن يتساءل عن قيمة A وهل هي صفرية أم لا ؟ وكذلك عن قيمة (B*B – 4.*A*C) وهل هي سالبة أم لا ؟ وماذا عليه أن يفعل في كل حالة ؟

ولنبدأ أولا بعمل مخطط تدفق لتلك المشكلة نحدد فيه ما يجب علينا عمله وماهي العمليات الحسابية التي يتطلب القيام بها ؟ فبفرض أن قيم C,B,A معروفة لدينا (سواء سبق حسابها أو قراءتها في البرنامج) ، فأن أول ماعلينا عمله هو اختبار قيمة المتغير A ومعرفة هل هي صفرية أم لا ؟

فاذا كانت A = Ø ، وجب علينا اعطاء الحاسب أمرا بعدم تنفيذ بقية البرنامج **والذهاب** إلى تعليمة **التوقف** .



شكل (٤-١)

- واذا كانت قيمة المتغير A غير صفرية ، وجب علينا أن نعرف قيمة المقدار $R = B^2 - 4AC$ وهل هي سالبة أم صفرية أم موجبة .

- فأذا كانت R سالبة (R<Ø) ، وجب علينا أيضا الذهاب الى تعليمة التوقف عن تنفيذ بقية البرنامج لأننا في هذه الحالة لن تتمكن من حساب قيمة الجذر التربيعي لكمية سالبة ، كما سبق أن أشرنا .

- واذا كانت قيمة R صفرية (R = Ø) ، فلن يكون هناك داع لجعل الحاسب يأخذ وقتاً في حساب الجذر التربيعي لكمية صفرية ، وخاصة اذا علمنا أن كل دقيقة على الحاسب تكلفنا الكثير ماديا . وبالتالي فيمكننا الاكتفاء لحساب قيمة Y بأن نجعل الحاسب يذهب الى التعليمة :

Y = -B / (2.*A)

- فاذا كانت قيمة R موجبة (R>Ø) فسنقوم بحساب قيمة Y بجعل الحاسب ينتقل الى التعليمة :

Y = (-B+R**Ø.5) / (2.*A)

مثـال ٢ :

– لنفرض أن المتغير Y يتم حسابه على النحو التالي :

i) $Y = (x-a)^3 - (x-a)^2 + (x-a) + 5.$

اذا كانت قيمة x اكبر من قيمة a . عندما تكون $(x>a)$.

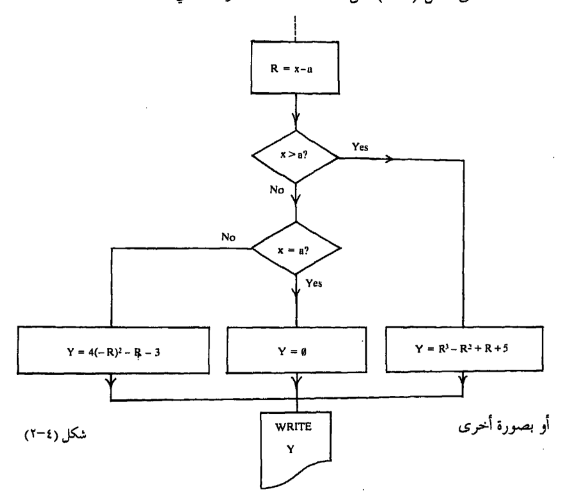
ii) $Y = 4\,(a-x)^2 + (a-x) - 3$

اذا كانت قيمة x اصغر من قيمة a . اي عندما تكون $(x<a)$

iii) $Y = \emptyset$

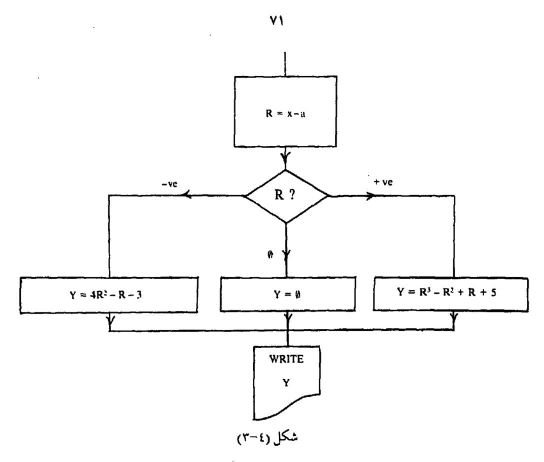
اذا كانت قيمة x تساوي قيمة a . أي عندما تكون $(x = a)$

فاذا افترضنا مرة أخرى أن قيم a,x معروفة لدينا (سواء سبق حسابها أو قراءتها في البرنامـج) ، فأن قيمة Y سيتوقف حسابها على قيم a,x وهل هما متساويتان أم أن أحداهما أكبر من الأخرى . ولذا فان مخطط التدفق شكل (٤–٢) لحل تلك المشكلة قد يكون كالتالي :



شكل (٤–٢)

أو بصورة أخرى



شكل (٤-٣)

من المثالين السابقين نلاحظ أننا قد نضطر في بعض الأحيان إن لم تكن أغلبها ، أن نقطع انسياق تسلسل تنفيذ التعليمات المتتالية في البرنامج ونتيجة الى تعليمة أخرى في موضع آخر في البرنامج قد تسبق أو تلي آخر تعليمة قام الحاسب بتنفيذها لتكون نقطة بدء مجموعة من التعليمات المتعاقبة يبدأ الحاسب في تنفيذها الواحدة تلو الأخرى الى أن يُواجَه مرة أخرى بتعليمة شرطية **اذا** ، تضطر الحاسب الى **الانتقال الى** موضع آخر في البرنامج وهكذا .

وفي لغة الفورتران فان ذلك يتم باستخدام تعليمات معينة تسمى تعليمات التحكم والانتقال Control and Branch Statements .

٤-١ تعليمة التحكم ..... اذا IF

هناك صورتين عامتين مختلفتين لهذه التعليمة في لغة الفورتران ، وان كانتا متشابهتين في عملهما . والصورة العامة الأولى لتعليمة التحكم ..... إذا IF هي :

٤-١-١ تعليمة اذا الحسابية :

nnnnn		IF (Arithmetic Expression) n_1, n_2, n_3

ويطلق على هذه التعليمة اسم **تعليمة اذا الحسابية**

حيث :

– nnnnn : رقم تعليمة اذا في البرنامج (ان وجدت)

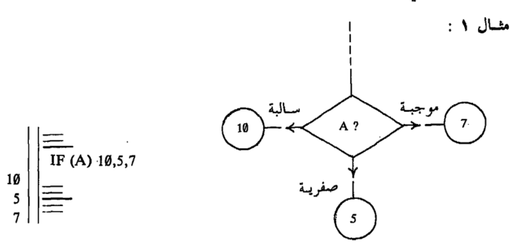
– Arithmetic Expression : ترمز الى تعبير رياضي مطلوب حساب قيمته ومعرفة ما اذا كان موجباً أم صفرياً أم سالباً . وهذا التعبير قد يكون متغيراً بسيطاً حقيقيا أو صحيحا أو تعبيرا رياضيا يحتوي على أكثر من متغير – بشرط أن تكون جميعها حقيقة أو صحيحة . **وليس** خليطاً منها Mixed mode .

– n_1 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي **سالبة** .

– n_2 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي **صفرية** ، أي تساوي صفراً .

– n_3 : رقم التعليمة التي تطلب فيها من الحاسب أن يذهب اليها في حالة ما اذا كانت قيمة التعبير الرياضي **موجبة** .

مثال ١ :



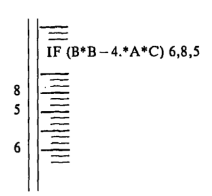
أي أنه يراد من الحاسب أن يذهب الى التعليمة التي رقمها 10 ، اذا كانت قيمة المتغير A سالبة ، وأن يذهب الى التعليمة التي رقمها 5 ، اذا كانت قيمة A تساوي صفرا ، وأن يذهب الى التعليمة التي رقمها 7 اذا كانت قيمة المتغير A موجبة .

نلاحظ من المثال السابق مايلي :

١ – التعبير الرياضي الموجود بين القوسين عبارة عن متغير بسيط قد يكون حقيقياً أو صحيحاً .

٢ – أرقام التعليمات لايلزم أن تكون مرتبة تصاعديا أو تنازليا .

مثـال ٢ :



أنه اذا كانت قيمة التعبير الرياضي الموجود مابين القوسين :

- سـالبة : فيجب الانتقال الى التعليمة التي رقمها 6 ،
- صفـراً : فيجب الانتقال الى التعليمة التي رقمها 8 ،
- موجبة : فيجب الانتقال الى التعليمة التي رقمها 5 ،

نلاحظ في هذا المثال مايلي :

١ – التعبير الرياضي الموجود بين القوسين عبارة عن عملية حسابية قد تحتوي على عمليات جمع وطرح وضرب وقسمة ... الخ .

٢ – عدم ظهور أرقام التعلميات في بقية البرنامـج بنفس ترتيب ظهورها في تعليمة اذا IF .

٣ – إن الحاسب سيقرر الانتقال الى التعليمة 6 أو التعليمة 8 أو التعليمة 5 ، بعد حساب التعبير الرياضي (B*B – 4.*A*C) ومعرفة قيمته .

مثـال ٣ :

```
    IF (A) 5,3,5
5   Y =  – B/(2.*A)
3
```

يتبين من هذا المثال أنه اذا كانت قيمة المتغير A سالبة أو موجبة (أي غير صفرية) ، فأن الحاسب سينتقل الى التعليمة التي رقمها 5 . أما اذا كانت قيمة المتغير A صفرية فسينتقل الحاسب الى التعليمة التي رقمها 3 . ولذلك فمن الممكن أن يتساوى رقمين في تعليمة التحكم اذا ، بينما يكون الرقم الثالث مختلفاً . وعلى سبيل المثال مايلي :

```
IF (S – 3.) 4,7,4
```

أي اذا كانت قيمة S تساوي 3 ، فعلى الحاسب أن يتوجه الى التعليمة التي رقمها 7 . أو بمعنى آخر ، اذا كانت قيمة S لاتساوي 3 ، فعلى الحاسب أن ينتقل الى التعليمة التي رقمها 4 .

مثـال :

```
IF (S – 3.) 6,6,8
```

أي أنه اذا كانت قيمة S اكبر من 3 ، (أي أن (S – 3) كمية موجبة) فعلى الحاسب أن ينتقل الى التعليمة التي رقمها 8 والا فيجب الانتقال الى التعليمة التي رقمها 6 .

```
IF (S – 3.) 4,2,2
```

أي أنه اذا كانت قيمة S اكبر من أو تساوي 3 ، فعلى الحاسب أن ينتقل الى التعليمة التي رقمها 2 . أو بمعنى آخر ، اذا كانت قيمة S أقل من 3 فأن الحاسب سينتقل الى التعليمة التي رقمها 4 .

٤–١–٢ تعليمة اذا المنطقية :

والصورة العامة الثانية لتعليمة التحكم .... اذا IF هي :

nnnnn	IF (Logical Expression) es

حيث :

- nnnnn : رقم تعليمة اذا ... IF في البرنامـج (ان وجدت) .
- Logical Expression : تعبير منطقي ينقسم في الحقيقة الى جزئين يراد المقارنة بينهما ، من حيث التساوي أو عدمه أو أن احدهما اكبر من الآخر أو أصغر منه . فأذا افترصنا أن جزئى التعبير المنطقي هما Exp1 , Exp2 فيمكن كتابة تعليمة اذا .. IF في أحدى الصور الست

التالية :

١ - IF (Exp1.EQ.Exp2) es

أي اذا كان (Exp 1. Equals يساوي Exp 2)

٢ - IF (Exp 1. GE. Exp 2) es

اي اذا كان (Exp 1. Greater or Equal اكبر من أو يساوي Exp 2)

٣ - IF (Exp 1 .GT. Exp 2) es

أي اذا كان (Exp 1 Greater Than أكبر من . Exp 2)

٤ - IF (Exp 1. LE. Exp 2)

أي اذا كان (Exp 1. Less or Equal أقل من أو يساوي . Exp. 2)

٥ - IF (Exp 1. LT. Exp 2) es

أي اذا كان (Exp 1. Less Than أقل من . Exp 2)

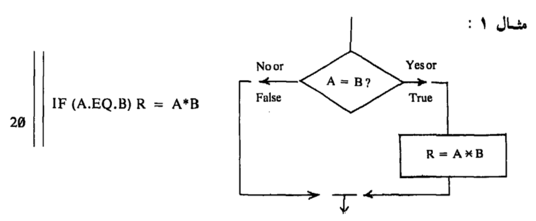
٦ - IF (Exp 1 .NE. Exp 2) es

أي اذا كان (Exp 1. Not Equalلايساوي . Exp. 2)

وقد يكون كل من التعبيرين Exp 1 , Exp 2 عبارة عن متغيرات بسيطة ، حقيقة أو صحيحة أو تعبيرات رياضية تحتوي على أكثر من متغير بشرط أن تكون جميعها من نوع واحد ، أما حقيقية أو صحيحة وليس خليطا منها .

- es : تعليمة تنفيذية Executable Statement يقوم الحاسب بتنفيذها اذا تحقق الشرط الذي يربط بين التعبيرين Exp 1 , Exp 2 واذا لم يتحقق ذلك الشرط فان الحاسب يقوم بتنفيذ التعليمة التي تلي تعليمة التحكم اذا .... IF مباشرة في البرنامج . ويطلق على هذه التعليمة اسم **تعليمة اذا المنطقية** Logical IF Statement .

مثال ١ :



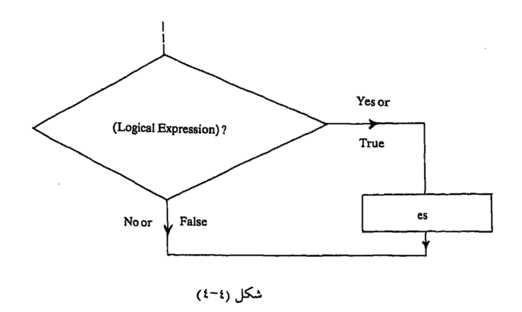
أي أنه اذا كانت قيمتى B,A متساويتين ، فان الحاسب سيقوم بحساب قيمة R ثم يبدأ في تنفيذ التعليمات التي تلي تعليمة اذا ... IF مباشرة أما اذا كانت قيمتى B,A مختلفتين ، فان الحاسب لن يقوم بحساب قيمة R بل سيتجه مباشرة الى التعليمة التالية والتي رقمها 2Ø .

مثـال ٢ :

```
IF ((B*B – 4.*A*C). GE.Ø.) Y = ( – B + (B*B – 4.*A*C)** Ø.5//(2.*A)
STOP
```

يتبين من هذا المثال أنه اذا كانت قيمة $(B^2 - 4\ AC)$ اكبر من أو تساوي الصفر ، فأن الحاسب سيقوم بتنفيذ بقية التعليمة ، أي حساب قيمة Y ثم يتوقف . أما اذا كانت قيمة $(B^2 - 4\ AC)$ سالبة ، أي أقل من الصفر فأن الحاسب سيتوقف دون حساب قيمة Y .

ومن المثالين السابقين يتبين أن تنفيذ بقية تعليمة اذا ... IF يتوقف على تحقق العلاقة المنطقية التي تربط بين جزئي التعبير المنطقي الموجود بين القوسين . كما يتبين أيضا أن الحاسب سيقوم بتنفيذ التعليمة التي تتلو تعليمة اذا ... IF المنطقية مباشرة ، بصرف النظر عن تنفيذ بقية التعليمة أم لا فيما عدا اذا كانت بقية تعليمة اذا هي احدى تعليمات الانتقال Branch Statements التي سنشرحها فيما بعد . وبذلك يمكن أن نتخيل رسم مخطط تدفق البيانات شكل (٤-٤) الخاص بتلك التعليمة كالتالي :



شكل (٤-٤)

٤-١-٣ ملحوظات على تعليمتى التحكم اذا .... IF

(أ) في تعليمة التحكم اذا ... IF الحسابية :

١ - لابد من كتابة التعبير الحسابي بين قوسين .

٢ - أن يكون التعبير الحسابي متجانس في تكوينه **وليس** خليطا من متغيرات صحيحة وحقيقية أو ثوابت متغيرة floating وثابته fixed .

٣ - في تعليمة **اذا** الحسابية يجب الالتزام بتسلسل أرقام التعليمات ، بحيث أن رقم التعليمة الأولى n_1 ستخصص كي ينتقل اليها الحاسب في حالة ما اذا كان التعبير الرياضي سالباً ، ورقم التعليمة الثانية n_2 اذا كانت قيمة التعبير الرياضي صفرا . ورقم التعليمة n_3 في حالة ام اذا كان التعبير الرياضي موجبا .

٤ - التحقق من وجود فاصلة Comma بين كل رقمى تعليمة .

٥ - أن أرقام التعليمات الثلاث قد تكون مختلفة أو تتساوى اثنتين منها .

٦ - أن أرقام التعليمات يجب أن تكون أعداداً صحيحة غير كسرية ولا تحتوي على اشارات سالبة أو موجبة .

٧ - الا تكون أرقام التعليمات متغيرات سواء حقيقية أو صحيحة ، حتى ولو سبق اعطاء قيم لتلك المتغيرات . فعلى سبيل المثال ، لايمكن القول :

```
      K = 5
      L = 7
      IF (          ) L,10,K
5     ...
7     ...
10    ...
```

```
      K = 5
      IF (e1.EQ.e2) GOTO K
```

(ب) في تعليمة التحكم اذا ... IF المنطقية :

١ - يجب أن يكون جزئى التعبير المنطقي Exp 1 ,Exp 2 المراد مقارنتهما من نوع واحد ، بحيث لا يجب أن يكون احدهما محتويا على متغيرات حقيقية والآخر على متغيرات صحيحة . وبالتالي أيضاً لايجب أن يحتوي أحدهما على خليط من المتغيرات الصحيحة والحقيقية .

٢ - التحقق من وجود رمز المقارنة بين جزئى التعبير المنطقي **بين نقطتين** .

٣ – سيقوم الحاسب بتنفيذ التعليمة التي تتلو تعليمة **اذا المنطقية** مباشرة ، مهما كانت نتيجة المقارنة بين جزئي التعبير المنطقي (Exp 2, Exp 1) **الا** اذا كانت التعليمة التنفيذية es المكملة لتعليمة إذا المنطقية هي تعليمة انتقال GOTO والتي سيأتي الحديث عنها في الفصل التالي .

الفصل الخامس
تعليمات الإنتقال

الفصــل الخـامـس

تعليمــات الإنتقــال Transfer Statements

مـقـدمـة : تعليمـة الإنتقـال ... GOTO

من تعليمات التحكم التي سبق شرحها ، تبين لنا أننا قد نضطر في بعض الأحيان الى قطع تسلسل تنفيذ مجموعة من التعليمات والاتجاه بالحاسب الى تنفيذ مجموعة أخرى من التعليمات في موضع آخر من البرنامـج قد يسبق موضع هذا القطع أو قد يأتي بعده . وفي جميع الأحوال كانت عملية قطع تسلسل تنفيذ تلك التعليمات والانتقال بالحاسب الى تنفيذ تعليمات أخرى في موضع آخر من البرنامـج ، تتم وفقا للتعليمة اذا IF التي تقوم باختبار إشارة كمية ما ، والتوجه بالحاسب طبقا لاشارة تلك الكمية .

وفي لغة الفورتران ، يمكن الانتقال أيضا بالحاسب من تنفيذ مجموعة تعليمات في البرنامـج الى مجموعة أخرى في مكان آخر من نفس البرنامج باستخدام تعليمات انتقال Branching Statements تنقسم الى نوعين :

٥-١ تعليمة الانتقال GOTO الغير مشروطة Un-conditional GOTO

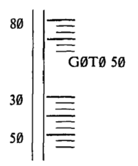
والصورة العامة لتلك التعليمة هي :

vvvvv	GOTO n

حيث :

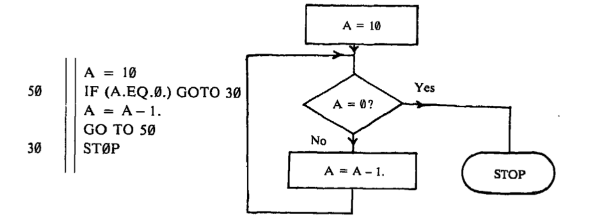
- n هي رقم التعليمة في البرنامـج ، والتي يراد من الحاسب الذهاب اليها لتنفيذها وتنفيذ التعليمات التي تليها ، ولابد أن تكون n رقما صحيحا موجبا وليس اسما لمتغير ، وقد تكون التعليمة التي رقمها n سابقة لتعليمة الانتقال أو لاحقة لها .
- vvvvv وهي رقم تعليمة الانتقال GOTO الغير مشروطة في البرنامج . وفي الحقيقة فان هذا الرقم ليس له داع في هذه التعليمة بالذات (أنظر ملحوظات على تعليمتى الانتقال) .

مثـال ٥-١-١ :



بعد أن يقوم الحاسب بتنفيذ التعليمة التي رقمها 80 ، يبدأ في تنفيذ التعليمات التي تليها مباشرة حتى يصل الى تعليمة الانتقال الغير مشروطة GOTO 50 والتي تتسبب في أن ينتقل الحاسب من هذا الموضع من البرنامـج الى الموضع الذي يبدأ بالتعليمة التي رقمها 50 دون تنفيذ التعليمة التي رقمها 30 والتعليمات التي تسبقها أو تليها .

مثـال ٥-١-٢ :



في المثال السابق نجد أن هناك تعليمتى انتقال احداهما الى التعليمة رقم 30 ، والأخرى الى التعليمة رقم 50 .

والتعليمة رقم 50 في هذا المثال تشمل تعليمتى التحكم والانتقال في آن واحد ، ولو أن تنفيذ الشق الثاني من هذه التعليمة (GOTO 30) سيتحقق عندما تصل قيمة A الى الصفر والتي عندها سينتقل الحاسب الى التعليمة التي رقمها 30 ، حيث يتوقف STOP عن تنفيذ بقية التعليمات التي تلي تلك التعليمة .

مثـال ٥-١-٣ :

بفرض أن هناك عدد N من الطلبة أدوا امتحانا في مادة ما ، وحسب درجة MARK كل منهم يراد معرفة :

١ - عدد الطلبة الذين حصلوا على درجة ممتاز (90 ≦ MARK ≦ 100)

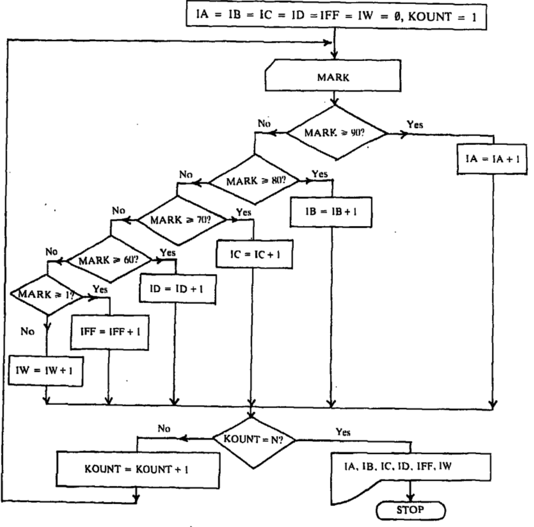
٢ - عدد الطلبة الذين حصلوا على درجة جيد جداً (80 ≦ MARK ≦ 89)

٣ - عدد الطلبة الذين حصلوا على درجة جيـد (70 ≦ MARK ≦ 79)

٤ - عدد الطلبة الذين حصلوا على درجة مقبول (60 ≦ MARK ≦ 69)

٥ - عدد الطلبة الراســبون (1 ≦ MARK ≦ 59)

٦ - عدد الطلبة المنسحبون (MARK = 0)



شكل (٥-١) مخطط التدفق للمثال أعلاه

بفرض أن عدد الطلبة من الفئة الأولى يمثلها المتغير IA ،
عدد الطلبة من الفئة الثانية يمثلها المتغير IB ،
عدد الطلبة من الفئة الثالثة يمثلها المتغير IC ،
عدد الطلبة من الفئة الرابعة يمثلها المتغير ID ،
عدد الطلبة من الفئة الخامسة يمثلها المتغير(*) IFF ،
عدد الطلبة من الفئة السادسة يمثلها المتغير IW ،

```
C        PUTING THE GRADE POSITIONS TO ZERO.
         IA = Ø
         IB = Ø
           :
         IW = Ø
C        STARTING COUNT THE NUMBER OF STUDENTS.
         KOUNT = 1
    5    READ (7, 1Ø) MARK
   1Ø    FORMAT (I3)
C        TESTING THE MARK AND PUTING IT IN THE APPROPRIATE
C        PLACE.
         IF (MARK. GE. 9Ø) GOTO 35
         IF (MARK. GE. 8Ø) GOTO 3Ø
         IF (MARK. GE. 7Ø) GOTO 25
         IF (MARK. GE. 6Ø) GOTO 2Ø
         IF (MARK. GE. 1) GOTO 15
         IW = IW + 1
         GOTO 1ØØ
```

(*) عند اختيار اسماء المتغيرات يتعين أن تكون تلك الأسماء مختلفة عن تعليمات الفورتران المعروفة مثل IF أو GOTO أو STOP أو ... الخ . كذلك يلاحظ أن جميع أسماء المتغيرات تبدأ بحرف I وذلك كي تكون قيم تلك المتغيرات صحيحة أي لاتقبل قيما كسرية . وهو مايتفق منطقيا مع ما تمثله تلك المتغيرات حيث أن كلا منها يمثل عددا للطلبة لايحتوي على قيم كسرية .

في مخطط التدفق السابق نلاحظ أننا بعد عمل الاختبارات المختلفة على درجة الطالب ووضعها في المكان المناسب لها ، نعاود قراءة الدرجة التي تليها . ويتحكم في تلك العملية متغيراً يعمل كعداد KOUNT يتزايد بعد قراءة كل درجة بمقدار الوحدة الى أن تصل قيمته الى N ، ويمكن تصور البرنامج الذي يعبر عن مخطط التدفق كما يلي :

```
   15 | IFF = IFF + 1
      | GOTO 100
   20 | ID = ID + 1
      | GOTO 100
   25 | IC = IC + 1
      | GOTO 100
   30 | IB = IB + 1
      | GOTO 100
   35 | IA = IA + 1
  100 | IF (KOUNT. EQ. N) GOTO 150
      | KOUNT = KOUNT + 1
      | GOTO 5
C     | WRITING THE RESULTS.
  150 | WRITE (7, 200) IA,IB,IC,ID,IFF,IW
  200 | FORMAT (6I8)
      | STOP
      | END
```

٥-٢ تعليمة الانتقال GOTO المشروطة أو (المحسوبة) : Computed GOTO

علمنا مما سبق أن تعليمة اذا IF الحسابية يمكن أن تنتقل بالحاسب الى ثلاث مواضع مختلفة على **الأكثر** من البرنامج (الانتقال الى موضع ما في البرنامج اذا كان المتغير أو التعبير سالبا والى موضع ثان عندما يكون المتغير أو التعبير قيمته صفرا والى ثالث عندما يكون موجبا) . كذلك فأن تعليمة الانتقال الغير مشروطة يمكن أن تنتقل بالحاسب الى موضع واحد فقط في البرنامج يبدأ بتعليمة لها رقم يتفق مع الرقم المصحوب بتعليمة الانتقال GOTO الغير مشروطة .

وهناك تعليمة انتقال أخرى يمكن باستخدامها التنقل الى اكثر من موضع في البرنامج ، ويطلق على هذه التعليمة اسم **تعليمة الانتقال GOTO المشروطة** . والصورة العامة لتلك التعليمة هي :

vvvvv	GOTO (n_1, n_2, n_3,...., n_m), iv

حيث :

- vvvvv : رقم تعليمة الانتقال GOTO المحسوبة في البرنامج ، وهذا الرقم **إختياري** يتوقف وجوده أوعدم وجوده على واضع البرنامج .

- $n_1, n_2, n_3, \ldots, n_m$: أرقام تعليمات لابد من تواجدها في البرنامج حيث قد يتطلب الأمر الانتقال اليها خلال تنفيذ الحاسب للبرنامج ، وبالطبع لابد أن تكون تلك الأرقام صحيحة ولا يحتوي على أية كسور . وقد تكون جميع هذه الأرقام مختلفة أو أن بعضها متشابه ، **ولابد من تواجد هذه الأرقام بين قوسين** وكذلك **لابد أيضا من تواجد فاصلة** Comma **بين كل رقم وآخر .**

- iv : متغير صحيح لابد أن يكون قد سبق اعطاؤه احدى القيم 1 أو 2 أو ... أو m . حيث أنه بناءا على قيمة ذلك المتغير سينتقل الحاسب الى التعليمة رقم n_1 أو n_2 أو ... أو n_m . كذلك يجب ملاحظة **وجود فاصلة** Comma قبل كتابة المتغير الصحيح .

وعندما يقوم الحاسب بتنفيذ تلك التعليمة ، فأنه ينتقل الى الجزء من البرنامج الذي يبدأ بالتعليمة التي رقمها n_1 اذا كنت قيمة التعبير iv هي 1 ، والى الجزء من البرنامج الذي يبدأ بالتعليمة التي رقمها n_2 اذا كانت قيمة المتغير iv هي 2 وهكذا . وبالتالي فاننا نتوقع أن قيمة المتغير iv في أي موضع من البرنامج لن يزيد عن m .

مثـال ٥-٢-١ :

```
     K = ...        - قيمة K التي ستأخذ أحدى القيم 1 أو 2 أو 3 أو 4
30
     GOTO (15, 10, 30, 10), K →   تعليمة الانتقال المحسوبة والتي
                                  بناء على قيمة K سيتوجه الحاسب الى التعليمة 15 أو 10 أو 30
10
```

تبعا لقيمة K . ففي هذا المثال اذا كانت قيمة K تساوي 1 ، فأنه عند تنفيذ الحاسب لتعليمة الانتقال المحسوبة سينتقل الى الموضع من البرنامج الذي يبدأ بالتعلمية رقم 15 .
واذا كانت K تساوي 2 فان الحاسب سينتقل الى التعليمة رقم 10 وهكذا ، ونلاحظ في هذا المثال :

١ - ان الحاسب سينتقل الى الوضع من البرنامج الذي يبدأ بالتعليمة رقم 10 عندما تأخذ K القيمة 2 أو 4 ، أي أن أرقام التعليمات بين القوسين قد تكون جميعها مختلفة أو بعضها متشابهة .

٢ - أن قيمة K في ذلك البرنامج لن تزيد عن 4 ، حيث لايوجد الا أربعة أرقام تعليمات بين القوسين حتى وأن تساوي فيهما اثنين أو اكثر .

٣ - قد تكون أرقام التعليمات الموجودة بين القوسين أرقاما لتعليمات تسبق أو تأتي بعد تعليمة الانتقال المحسوبة نفسها .

مثـال ٥-٢-٢ :

باستخدام تعليمة الانتقال GOTO المحسوبة ، اكتب برنامجا لحساب .

١ - مجموع الأعداد الفردية الواقعة بين 1 و N (1+3+5+...+N)

٢ - مجموع الأعداد الزوجية الواقعة بين 1 و N (2+4+6+...+N)

بأفتراض أن مجموع الأعداد الفردية يمثله المتغير (Sum of ODD nos.) SODD

وأن مجموع الأعداد الزوجية يمثله المتغير (Sum of EVEN nos.) SEVEN

ولاجراء أي عمليات تجميع في أي متغير يلزم أن نضع في هذا المتغير القيمة صفرا .

أي نضع SODD = Ø ، SEVEN = Ø وتكمل البرنامج كالتالي :

```
        L = 1 →                     عداد يبدأ بالقيمة 1
        SEVEN = Ø. →                تصفير المتغير الذي سيتم فيه تجميع الأعداد الزوجية
        SODD = Ø. →                 تصفير المتغير الذي سيتم فيه تجميع الأعداد الفردية
        K = 1 →                     متغير للتحكم في تعليمة الانتقال GOTO المحسوبة
100     GOTO (1Ø, 2Ø), K
1Ø      SODD = SODD + FLOAT (L)
        GOTO 3Ø
2Ø      SEVEN = SEVEN + FLOAT (L)
        K = Ø
3Ø      IF (L. EQ. N) GOTO 15Ø
        L = L + 1
        K = K + 1
        GOTO 1ØØ

15Ø     WRITE (7, 2ØØ) SODD, SEVEN
2ØØ     FORMAT (1X, 2f14.)
        STOP
        END
```

مثـال ٥-٢-٣ :

في امتحان ما ، تم عمل ترقيم لمجموعة عددها N من الطلبة والطالبات بحيث أن لكل طالب أو طالبة رقمين ، أحدهما للتعريف بالنوع (ذكر أو أنثى) بحيث يخصص الرقم 1 للذكور والرقم 2 للأناث . والأخر للتعريف بتقدير الطالب أو الطالبة بحيث يخصص :

الرقم 1 لكل طالب أو طالبة حاصل على درجة امتياز ،

الرقم 2 لكل طالب أو طالبة حاصل على درجة جيد جدا ،
الرقم 3 لكل طالب أو طالبة حاصل على درجة جيد ،
الرقم 4 لكل طالب أو طالبة حاصل على درجة مقبول ،
الرقم 5 لكل طالب أو طالبة راسب في الامتحان ،
الرقم 6 لكل طالب أو طالبة منسحب من الامتحان .

والمطلوب حساب :

١ – عدد الطلبة الذكور .
٢ – عدد الطلبة الاناث .
٣ – عدد الطلبة والطالبات في كل درجة من الدرجات الست السابقة .

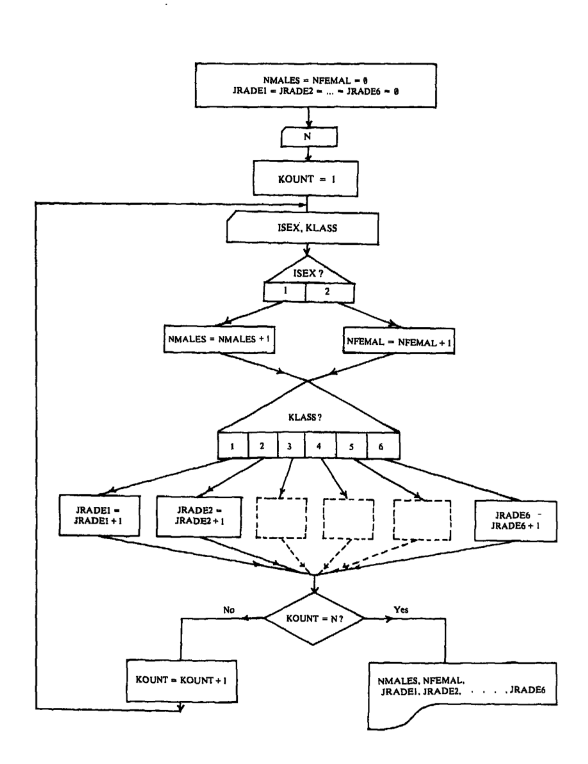
بافتراض أن عدد الطلبة الذكور سيمثله المتغير (Number of MALES) NMALES
وأن عدد الطلبة الاناث سيمثله المتغير (Number of FEMALES) NFEMAL
وأن عدد الطلبة والطالبات في كل درجة من الدرجات الست الباقية يمثلها المتغيرات :

JRADE 1 مجموع الطلبة والطالبات الذين حصلوا على تقدير امتياز ،
JRADE 2 مجموع الطلبة والطالبات الذين حصلوا على تقدير جيد جدا ،
JRADE 3 مجموع الطلبة والطالبات الذين حصلوا على تقدير جيد ،
JRADE 4 مجموع الطلبة والطالبات الذين حصلوا على تقدير مقبول ،
JRADE 5 مجموع الطلبة والطالبات الذين رسبوا في الامتحان ،
JRADE 6 مجموع الطلبة والطالبات الذين انسحبوا من الامتحان .

واذا فكرنا فيم يجب علينا اعطاؤه للحاسب كي نتمكن من حساب ماهو مطلوب منا ، نجد أن هناك قيمتين لابد أن يعرفهما الحاسب لكل طالب أو طالبة :

١ – متغير ولنطلق عليه اسم ISEX يأخذ القيمة 1 أو 2 ليبين للحاسب النوع (طالب أم طالبة) ،
٢ – متغير ولنطلق عليه اسم KLASS يأخذ القيمة من 1 حتى 6 ليبين للحاسب الدرجة التي حصل عليها ذلك الطالب (أو الطالبة) .

ولذا يمكننا أن نتخيل مخطط التدفق لحل تلك المشكلة كالتالي :



شكل (٥-٢) مخطط التدفق للمثال أعلاه

وبذا يمكن كتابة البرنامج كالتالي :

```
C        PROGRAM 1 TO CALCULATE THE NO. OF STUDENTS,
C        (MALES, FEMALES)
         NMALES = Ø
         NFEMAL = Ø
         JRADE 1 = Ø
         JRADE 2 = Ø                          تصفير المتغيرات التي سيتم فيها عملية التجميع
         JRADE 3 = Ø
         JRADE 4 = Ø
         JRADE 5 = Ø
         JRADE 6 = Ø
         READ (7, 1ØØ) N
  1ØØ    FORMAT (I8)
         KOUNT = 1                            عداد يحسب من 1 الى N
    1    READ (7,5) ISEX, KLASS               قراءة بيان أول طالب أو طالبة وهذا البيان
    5    FORMAT (2I3)                         يحتوي على النوع والدرجة
         GOTO (1Ø, 15), ISEX                  توجيه الحسابات حسب قيمة المتغير ISEX
   1Ø    NMALES = NMALES + 1                  اضافة طالب الى عدد الطلبة الذكور
         GOTO 2Ø
   15    NFEMAL = NFEMAL + 1                  اضافة طالبة الى عدد الطالبات الاناث
   2Ø    GOTO (25,3Ø, 35,4Ø, 45, 5Ø), KLASS
   25    JRADE 1 = JRADE 1 + 1   اضافة 1 الى عدد الطلبة والطالبات الحاصلون على امتياز
         GOTO 55
   3Ø    JRADE2 = JRADE2 + 1     اضافة 1 عدد الطلبة والطالبات الحاصلون على جيد جدا
         GOTO 55
   35    JRADE3 = JRADE3 + 1     اضافة 1 عدد الطلبة والطالبات الحاصلون على جيد
         GOTO 55
   4Ø    JRADE4 = JRADE4 + 1     اضافة 1 عدد الطلبة والطالبات الحاصلون مقبول
         GOTO 55
   45    JRADE5 = JRADE5 + 1     اضافة 1 الى عدد الطلبة والطالبات الراسبون
         GOTO 55
   5Ø    JRADE6 = JRADE6 + 1     اضافة 1 الى عدد الطلبة والطالبات المنسحبون
   55    IF (KO UNT. EQ. N) GOTO 6Ø
         KOUNT = KOUNT + 1
         GOTO 1
   6Ø    WRITE (7,65) NMALES, NFEMAL, JRADE1, JRADE2,
        1JRADE3, JRADE4, JRADE5, JRADE6
   65    FORMAT (8I6)
         STOP
         END
```

وقد يتساءل البعض عن امكانية تنفيذ البرنامج السابق باستخدام تعليمة اذا .. IF المنطقية ، وفي هذه الحالة ماهو الفرق بين الطريقتين ؟ والاجابة على ذلك نعم كما يلي :

```
C      PROGRAM 2 OF EXAMPLE5 - 2 - 3
       NMALES = 0
       :    :    :
       JRADE6 = 0                              كما في برنامج 1
       .    .    .
       KOUNT = 1
1      READ (7,5) ISEX, KLASS
5      FORMAT (2I3)
       IF (ISEX. EQ. 1) GOTO 10
       NFEMAL = NFEMAL + 1
       GOTO 15
10     NMALES = NMALES + 1
15     IF (KLASS. EQ. 1) GOTO 20
       IF (KLASS. EQ. 2) GOTO 25
       IF (KLASS. EQ. 3) GOTO 30
       IF (KLASS. EQ. 4) GOTO 35
       IF (KLASS. EQ. 5) GOTO 40
       JRADE6 = JRADE6 + 1
       GOTO 55
20     JRADE1 = JRADE1 + 1
       GOTO 55
25     JRADE2 = JRADE2 + 1
       GOTO 55
30     JRADE3 = JRADE3 = 1
       GOTO 55
35     JRADE4 = JRADE4 + 1
       GOTO 55
40     JRADE5 = JRADE5 + 1
55     IF (KOUNT. EQ. N) GOTO 60
          :        :
          :        :                           كما في برنامج 1
       END
```

وقد يتبين مما سبق أننا إستبدلنا تعليمة الانتقال GOTO المحسوبة بمجموعة متتالية من تعليمة اذا .. IF المنطقية ، وبالطبع ستكون نتائج البرنامجين متطابقة ، ولكن قد يختلف زمن تنفيذ كل برنامج عن الآخر حيث يتوقف وقت تنفيذ أي برنامج على التعليمات النهائية الخاصة بلغة الحاسب (Machine Language) .

تمرينات على تعليمتي الانتقال GO TO ، اذا IF .

١ – أوجد القيمة النهائية للمتغير M التي نحصل عليها بعد الانتهاء من تنفيذ اجزاء البرامج التالية ، اذا كانت قيمة M في البداية هي 3 وقيمة N هي 6 :

i)
```
      IF (3*M. EQ. N) M=M+2
      M=M+3
```

ii)
```
      IF (N. GT. M) GO TO 30
      M=M+1
      GO TO 20
30    M=N
20    M=M-N
```

iii)
```
      IF (N-M) 3,6,3
6     M=M+1
3     M=N
```

iv)
```
      IF (3*M-2*N) 2,5,5
2     M=N
5     M=M+1
```

v)
```
      IF (2*M. LT. N) M=M-2
      M=M+2
```

vi)
```
      IF (3*N. LE. 2*M) GO TO 10
      M=M+1
      GO TO 20
10    M=N
20    M=M+N
```

٢ – أوجد تعليمات الانتقال المحسوبة الغير صحيحة في التعليمات التالية ، وبين سبب الخطأ .

i) GO TO (1Ø,5,8,Ø,15), L

ii) GO TO (7,11,11,11,9,7), K

iii) GO TO (4,5.,9), Y

iv) GO TO (8,1Ø,3,6) M

٣ – اكتب تعليمة اذا IF الحسابية التي تكافىء التعليمة التالية :

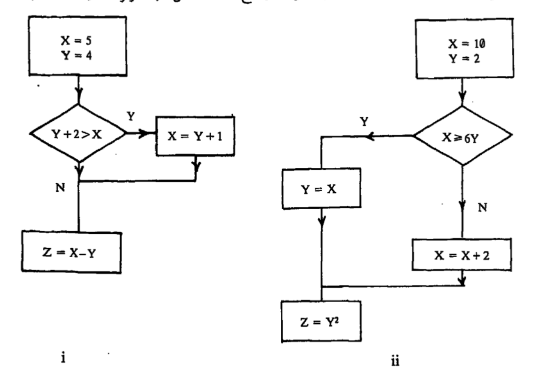
i) GO TO (15,1Ø,3Ø), L

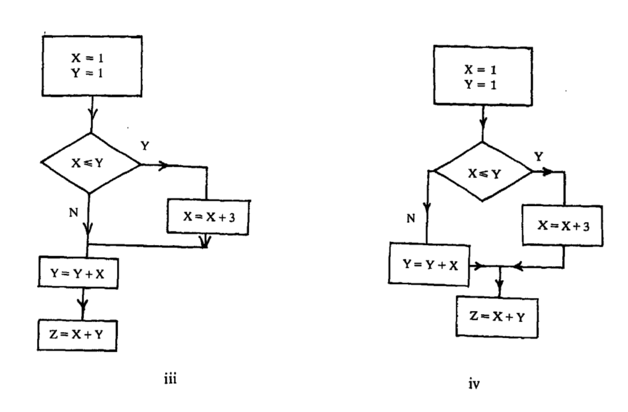
ii) GO TO (22,55,33,77,55), KOKO

٤ – في تعليمات اذا IF التالية توجد أخطاء اما في التعليمة نفسها أو عند تنفيذها ، صحح تلك الأخطاء :

i)	IF (I = N), 4,1Ø,5	ii)	IF (X + Y – 4) 5,5,7
iii)	IF X – Ø 4,5,6	iv)	IF (X + Y*Y) 5,2
v)	1Ø | | IF (X + Y) 5,5,1Ø	vi)	16 | | IF (A – (B + C) 3, 1Ø,5Ø
vii)	IF (X + 4) 3,7	viii)	3 | | IF (A*N) 8,3,6

٥ – في مخططات التدفق التالية ، اكتب اجزاء البرامج المناسبة لكل منها ، وأوجد قيمة Z الناتجة :





٦ – عندما تكون قيمة المتغير X بين صفر وواحد فأن قيمة المتغير Y تساوي X^2 ، عندما تكون قيمة المتغير X بين ثلاثة وأربعة فأن قيمة المتغير Y تساوي (X –) ، اذا كانت قيمة X غير ذلك فأن Y تساوي صفرا .

إرسم مخطط تدفق لتلك المشكلة وأكتب الجزء من البرنامج المناسب .

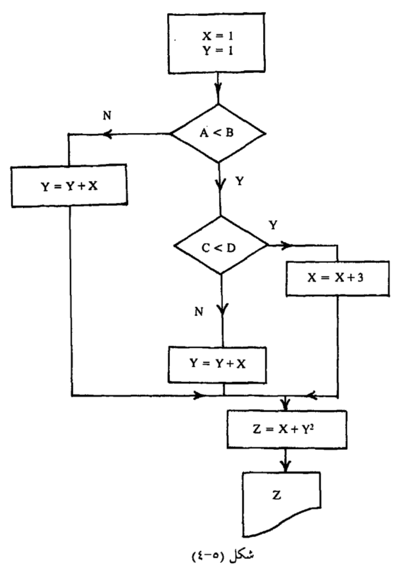
٧ – اكتب الجزء من البرنامج الذي يناسب مخطط التدفق التالي شكل (٥–٤) ، وأحسب قيمة Z في الحالات التالية :

1) A = 2 , B = 3 , C = 3 , D = 2

2) A = 2 , B = 3 , C = 2 , D = 3

3) A = 3 , B = 2 , C = 2 , D = 3

4) A = 3 , B = 2 , C = 3 , D = 2



شكل (٥-٤)

٩ - أرسم مخطط تدفق واكتب الجزء من البرنامج لحساب :

$$S = \sum_{i=1}^{N} X_i Y_i$$

١٠ - من المعروف نظريا أن مجموع أي ضلعين في مثلث اكبر من الضلع الثالث فأذا أعطيت ٦٠ قيمة متتالية ، كل ثلاث منها تمثل أطوالا لأضلاع قد تكون مثلثا أو لاتكون . اكتب برنامج يقرأ تلك القيم كل ثلاثة على انفراد ثم يبين إن كانت تلك القيم تمثل مثلثا أم لا .

(ملحوظة) C, B, A تكون مثلثا اذا كان $B + C > A; A + C > B; A + B > C$

١١ – من المعروف نظريا أن جذري معادلة من الدرجة الثانية : $AX^2 + BX + C = 0$.
يمكن حسابها من القانون :

$$X_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A} \quad ; \quad X_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

بحيث أنه اذا كانت :

(أ) $(B^2 - 4AC)$ اكبر من الصفر ، فأن الجذرين يكونان حقيقيين .

(ب) $(B^2 - 4AC)$ أقل من الصفر ، فأن الجذرين يكونان تخيليين .

(جـ) $(B^2 - 4AC)$ تساوي صفرا ، فأن الجذرين يكونان متساويين وكل منهما يساوي : $\frac{-B}{2A}$

(د) A تساوي صفرا فانه لا توجد جذور للمعادلة .

أرسم مخطط تدفق ، ثم اكتب الجزء من البرنامج المناسب .

١٢ – اكتب برنامج لحساب :

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \dots + \frac{99}{100}$$

١٣– اكتب برنامج لحساب :

$$P = \frac{1}{1^2} \cdot \frac{3}{2^2} \cdot \frac{5}{3^2} \qquad \frac{2N-1}{N^2}$$

١٤– اكتب برنامج لحساب :

$$S = 1 - \quad \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots \pm \frac{1}{N}$$

١٥– في أحد المحال التجارية ، يتم عمل خصم على المشتروات طبقاً للشروط التالية :

(أ) اذا كانت قيمة المشتروات أقل من ٢٠٠ ريالاً ، فلن يكون هناك خصما ،

(ب) اذا كانت قيمة المشتروات أكبر من أو يساوي ٢٠٠ ريالاً وأقل من أو يساوي ٥٠٠ ريالاً فأن قيمة الخصم تساوي ١٠٪ من قيمة المشتروات ،

(جـ) اذا كانت قيمة المشتروات أكبر من ٥٠٠ ريالاً ، فأن قيمة الخصم تساوي (٥٠+٨٪) من المتبقي من قيمة المشتروات بعد خصم ٥٠٠ ريالاً منها .

أرسم مخطط تدفق وأكتب البرنامج المناسب .

١٦– في أحدى الجامعات تم عمل احصاء للطلبة والطالبات الملتحقين بها ، وذلك بعمل بطاقة لكل طالب أو طالبة مثقب بها البيانات التالية :

السن – النوع (١ للطالبة ، ٢ للطالب) – المستوى الدارسي (١ : للمستوى الأول ، ٢ : للمستوى الثاني ، . . . ، ٥ : للمستوى الخامس ، ٦ : لغير المقيدين نظامياً) – الحالة العائلية (١ : للأعزب ، ٢ : للمتزوج) .

يتم الكشف عن نهاية البطاقات بوضع بطاقة بها قيمة تساوي ٣ في خانة النوع .

أرسم مخطط تدفق واكتب البرنامج المناسب لحساب :

(أ) النسبة المئوية للطلبة ،

(ب) النسبة المئوية للطالبات ،

(جـ) النسبة المئوية للطلبة والطالبات في المستويات الخمس المختلفة .

(د) النسبة المئوية للطلبة والطالبات الذين تزيد أعمارهم عن ٣٠ سنة .

(هـ) النسبة المئوية للمتزوجون .

١٧– اكتب برنامج يقرأ ثلاث قيم صحيحة ويقارن بينها ، بحيث اذا :

(أ) لم تكن هناك قيمتين متساويتين ، يكتب الرقم Ø

(ب) كانت هناك قيمتين متساويتين ، يكتب ترتيبهما (االأول والثالث مثلاً يكتب 3 – 1) .

(جـ) كانت القيم الثلاث متساوية ، يكتب 3 .

أختبر البرنامج على الحالات التالية : 787 , 998 , 979 , 133 , 666

١٨– مجموعة من البطاقات مثقب في كل منها رقماً صحيحاً موجباً (أقل من ١٠٠٠) ، وفي نهاية تلك المجموعة من البطاقات بطاقة لتعريف الحاسب بنهاية المجموعة .

(أ) أكتب برنامجاً لايجاد أكبر رقم في تلك المجموعة .

(ب) أكتب برنامجاً لايجاد أصغر رقم في تلك المجموعة .

١٩– بإفتراض أن تعداد السكان في الدولتين أ ، ب هما ٥٢ ، ٨٥ مليون نسمة على التوالي ، وكان معدل نمو السكان بهما هي ٤٪ ، ٢٪ على التوالي .أكتب برنامج لأعطاء تعداد السكان (لأقرب ١٠٠٠) في كل من الدولتين كل سنة أن يزيد تعداد السكان في الدولة أ عن الدولة ب . وأوجد عدد السنوات التي يتحقق فيها ذلك .

الفصـــل الســـادس

تعليمة حلقة التنفيذ المتكرر

الفصــل الســادس

تعلمية حلقة التنفيذ المتكرر DO loop

مقـدمـة : "Every Program can be shortened"

أي « كل برنامـج يمكن تصغيره أو إختصاره » ، شعار ظل ماثلا ولن ينسى منذ بدأنا خطواتنا الأولى في تعلم لغة الفورتران ، وبالطبع فهذا الشعار لايخص مستعملي لغة الفورتران فقط ولكن يمكن تطبيقه على أي لغة أخرى من لغات الحاسب ولكن العبرة هو في تطبيق الشعار الثاني "THINK" أي فكّر وهو أحد الشعارات وأهمها التي تضعها شركات IBM على مطبوعاتها وملصقاتها .

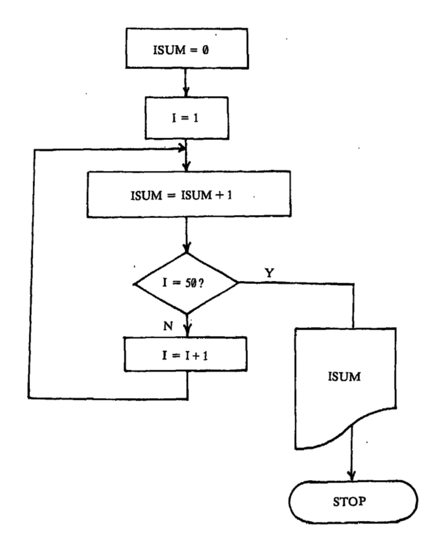
فأي مشكلة حسابية يمكن لحلها كتابة أكثر من برنامـج وجميعها ستعطى نتائـج أن لم تكن متفقة تماماً مع بعضها ، فستكون متقاربة(*) وهناك مشاكل خاصة تتطلب دقة كبيرة في حساباتها مثل المشاكل التي تتعلق بتجارب معملية فيزيائية أو كيميائية أو فلكية أو ... الخ ، ولكن معظم المشاكل لاتتطلب تلك الدقة الكبيرة في نتائجها ، فلا أقل من أن نراعي في حلها السهولة والبساطة والاختصار خاصة في كتابة برنامج ذلك الحل .

وأحدى الوسائل الفعالة في اختصار معظم البرامـج المكتوبة بلغة الفورتران هو إستخدام حلقات التنفيذ المتكررة ، والتي تختص بتنفيذ مجموعة تعليمات معينة في البرنامـج ، عدد معين من المرات بصفة تكرارية . ولنفترض المثال البسيط التالي والذي يراد فيه بإستخدام تعليمات الفورتران التي سبق دراستها حتى الآن ، ايجاد مجموع الأعداد التي تقع بين العددين ١ ، ٥٠ أي أن المطلوب ايجاد قيمة :

$$\sum_{i=1}^{5\emptyset} i = 1 + 2 + 3 + \dots + 5\emptyset$$

(*) قد تكون هناك فوارق في نتائـج حل مشكلة ما ، نظراً لما تتعرض له الحسابات من نوعين مختلفين من الأخطاء :
(أ) اخطاء حذف التقريب الدائري Round off error : وهو بأن نقوم بحذف الأرقام ذات الأهمية الصغرى في النتائج الحسابية ، ثم تقرب النتيجة الى عدد معين من الأرقام العشرية .
(ب) اخطاء الانهاء المبكر Truncation errors : وهو بأن تقوم بانهاء العمليات الحسابية بأخذ حدود معينة وإهمال بقية تلك الحدود توفيراً للوقت والجهد . فالمقدار: $\frac{1}{3} \neq 0.3 \neq 0.33 \neq \dots$

فأذا افترضنا أن نتيجة الجمع سيتم تخزينها في المتغير ISUM ، فأننا يمكننا ايجاد حاصل الجمع طبقاً لمخطط تدفق البيانات التالي :



شكل (٦-١)

ويكون البرنامج كالتالي :

```
      ISUM = Ø ............................ وضع قيمة المتغير الذي سيتم فيه التجميع بالقيمة صفر
      I = 1 ................................................ عداد سيأخذ القيم 3,2,1 , ... , 50
5     ISUM = ISUM + I .................... القيمة الجديدة للمتغير ISUM تساوي القيمة
                                          السابقة لنفس المتغير مضافاً اليها قيمة المتغير I
      IF (I.EQ.5Ø) GOTO 1Ø
      I = I + 1 ................................. طالما أن قيمة المتغير I لم تصل بعد الى القيمة 5Ø
                                          فسيزيد قيمة ذلك المتغير بواحد صحيح ....
      GOTO 5 ....................................... الانتقال الى التعليمة رقم 5 في البرنامج وتكرار
                                          تنفيذ الخطوات التي تليها ....
1Ø    WRITE (6,15) ISUM ........................ كتابة قيمة المتغير الذي تمت فيه عملية التجميع
15    FORMAT (I6)
      STOP
      END
```

نلاحظ في المثال السابق مايلي :

١ – أن مجموعة التعليمات التي تبدأ بالتعليمة رقم 5 وتنتهي بالتعليمة GOTO 5 تكوّن فيما بينها حلقة (loop) ، يتكرر تنفيذها عددا معينا من المرات .

٢ – يتوقف تكرار تنفيذ تلك الحلقة على قيمة المتغير I (العداد) ، فأن كانت قيمة I لاتساوي 5Ø ، أي لم يصل الى القيمة النهائية المطلوب الحساب عندها فأن الحاسب سيقوم بتنفيذ تلك الحلقة مرة أخرى وهكذا حتى تصل قيمة المتغير I الى القيمة النهائية المطلوب التوقف عندها .

سؤال : ماذا يحدث اذا استبدلنا التعليمة رقم 5 بالتعليمة التي تليها أي بدّلنا هاتين التعليمتين بحيث يصير البرنامج السابق كالتالي :

```
      ISUM = Ø
      I = 1
5     IF (I.EQ.5Ø)GOTO 1Ø
      ISUM = ISUM + I
      I = I + 1
      GOTO 5
1Ø    WRITE (6,15) ISUM
15    FORMAT (I6)
      STOP
      END
```

هل سنصل الى نفس النتيجة ؟ واذا كان هناك اختلافا في النتيجتين فما هو السبب وكيف يمكن تصحيحه ؟

٣ – أن المتغير I يبدأ بقيمة معينة يمكن أن نسميها القيمة الابتدائية Initial value or starting value أي التي سيبدأ العد بها ، وينتهي بقيمة معينة يمكن أن نسميها القيمة النهائية Final value .

٤ – أن المتغير I والذي يمثل عدادا في هذا المثال ، يستخدم أيضاً كمتغير يمكن استخدامه داخل بعض التعليمات التي تحتوي عليها الحلقة (ISUM = ISUM + I) .

٥ – عند تكرار تنفيذ تعليمات تلك الحلقة فأن المتغير I يزداد في كل مرة بمقدار الوحدة ولكن زيادة قيمة المتغير I تتم عن طريق تعليمة داخل الحلقة نفسها (I = I+1) ولذا فيمكن تغيير تلك التعليمة بحيث يزداد المتغير I بأي مقدار صحيح موجب .

(**سؤال** : هل يمكن تحوير البرنامج السابق بحيث يقوم بجمع قيم الأعداد الفردية الواقعة بين 1 , 50 ؟)

٦ – طالما أن المتغير I يستخدم كعداد ليحسب عدد مرات تنفيذ التعليمات الموجودة داخل تلك الحلقة ، فيجب أن نتوقع أن :

(أ) أول قيمة سيأخذها المتغير I ستكون أقل من القيمة النهائية .. واذا تساوت القيمتين فأن الحاسب سيقوم بتنفيذ التعليمات الموجودة داخل الحلقة مرة واحدة فقط .

(ب) أن المتغير I سيزداد بقيم صحيحة موجبة .

في المثال البسيط السابق ، عرفنا كيف يمكن للحاسب أن يقوم بتنفيذ مجموعة من التعليمات تكون فيما بينها حلقة loop . وفي الحقيقة فكثيراً ما نواجه بعض المشكلات التي يتطلب لتنفيذها على الحاسب عمل برامج تحوي على مثل تلك الحلقات المتكررة ، وفي كل حلقة منها لابد من تبليغ الحاسب بزيادة مقدار المتغير الذي يمثل العداد بقيمة ما ، وفي كل مرة عند تنفيذ كل حلقة لابد من أن نجعل الحاسب يسأل عن قيمة المتغير الذي يمثل العداد وهل وصل الى القيمة النهائية المطلوبة أم لا ؟ وبالطبع فأن مثل تلك التعليمات الممثلة في زيادة قيمة المتغير الذي يمثل العداد وكذلك في عمل مقارنة في كل مرة بين القيمة التي وصل اليها العداد والقيمة النهائية وكذلك في عملية الانتقال الى بداية الحلقة في كل مرة يتكرر فيها تنفيذ مجموعة التعلميات التي تحويها ، ستستغرق وقتاً كبيراً نسبياً من الحاسب حيث سيتوقف هذا الوقت على عدد مرات تنفيذ التعليمات التي تحويها الحلقة وعلى عدد التعليمات التي توجد داخل تلك الحلقة ، كما ستزيد من حجم البرنامج نفسه .

ولذا فأن لغة الفورتران تحوي تعليمة خاصة وهامة يمكن ويستحسن إستخدامها عندما يتطلب الأمر تنفيذ مجموعة التعليمات بصفة تكرارية لعدد معين من المرات ، وهذه التعليمة هي :

٦-١ الصورة العامة لتعليمة حلقة التنفيذ المتكرر DO Statement (loop)

وتكتب في الصورة العامة كالتالي :

vvvvv	$DO\ n\ i = i_I, i_F, i_S$

حيث :

vvvvv : رقم بدء تعليمة حلقة التنفيذ المتكررة ، وهو رقم إختياري يتوقف وجوده أو عدم وجوده على واضع البرنامج .

n : رقم نهاية تعليمة حلقة التنفيذ المتكرر ، ولابد من تواجده في البرنامج .

i : اسم متغير صحيح Integer variable يستخدم كعداد في حلقة التنفيذ المتكرر وقد يستخدم كمتغير صحيح داخل بعض تعليمات الحلقة نفسها ، ويطلق عليه اسم Index .

i_I : القيمة الابتدائية (أول قيمة Initial value) للمتغير i .

i_F : القيمة النهائية (آخر قيمة Final Value) للمتغير i .

i_S : القيمة التي سيزداد بها المتغير i في بدء تكرار تنفيذ مجموعة التعليمات التي تحويها الحلقة . ويطلق عليها اسم (الخطوة Increment or Step) .

والقيم الثلاث i_S , i_F , i_I لابد أن تكون قيماً عددية صحيحة موجبة أو أسماء متغيرات صحيحة **سبق** اعطاؤها قيماً عددية صحيحة موجبة في البرنامج .

وإذا كان المتغير الصحيح i يزداد بمقدار الوحدة (أي أن $i_S = 1$) فأن الصورة العامة تكون كالتالي :

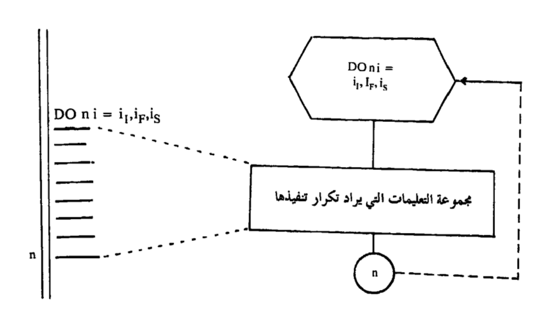
vvvvv	$DO\ n\ i = i_I, i_F$

وفي كلتا الصورتين يجب :

١ – وضع اسم المتغير الصحيح i على يسار علامة التساوي ، بينما يوضع على يمينها قيمة i_I أولاً ، ثم قيمة i_F ثانياً ، ثم قيمة i_S (أن تطلب الأمر ذلك) . كما يجب التأكد من وجود الفاصلة Comma بين كل قيمة وأخرى .

٢ – أن تأتي التعليمة التي رقمها n والتي تمثل تعليمة نهاية حلقة التنفيذ بعد (وليس قبل) بدء تعليمة حلقة التنفيذ المتكرر .

وتمثل حلقات التنفيذ في مخططات التدفق بالرمز ⬡ وبذا يمكن رسم مخطط التدفق لتعليمة حلقة التنفيذ المتكرر كالتالي :




```
10 || DO 10 L = 1,100,5
```

مثـال ٦-١-١ :

في هذا المثال سنجد أن الحاسب يقوم بتنفيذ مجموعة التعليمات التي تمثل حلقة التنفيذ المتكرر كالتالي :

- يأخذ قيمة المتغير $L = 1$ وينفذ مجموعة التعليمات التي تلي تعليمة حلقة التنفيذ المتكرر DO الى أن يصل الى التعليمة التي رقمها 10 والتي تمثل آخر تعليمة في الحلقة .
- ثم يأخذ قيمة المتغير $L = 1+5 = 6$ وينفذ مجموعة التعليمات حتى يصل الى التعليمة 10 .
- يأخذ قيمة المتغير $L = 6+5 = 11$ بعد ذلك ويستمر كما سبق الى أن يصل الى قيمة L النهائية والتي لابد أن تكون **أقل من أو تساوي** القيمة النهائية 100 ، أي إلى أن يصل إلى قيمة : $L = 91+5 = 96$.

نلاحظ في هذا المثال :

- عند مقارنة تعليمة حلقة التنفيذ في هذا المثال بالصورة العامة سنجد أن :

$n = 10 , i = L, i_I = 1, i_F = 100 , i_S = 5$

– إن المتغير L والذي يمثل العداد في هذا المثال لن يتزايد بمقدار الوحدة عند تكرار تنفيذ التعليمات التي تمثل حلقة التنفيذ بل سيأخذ القيم :

$1,\ 1+5 = 6,\ 6+5 = 11,\ 11+5 = 16,\ 16+5 = 21,\ \ldots,\ 91+5 = 96$

وهذا ماكنا نعبر عنه في السابق بالتعليمة : $L = L+5$.

– يستمر الحاسب في تكرار تنفيذ التعليمات التي تحويها حلقة التنفيذ المتكرر ولا يخرج من هذه الحلقة الا بعد أن تصل قيمة المتغير L إلى 101 (5+96) أي عندما تكون قيمة L أكبر من القيمة النهائية i_F .

مثـال ٦–١–٢ :

```
      DO 5 K = 3,7
 5    ═══
```

نلاحظ في هذا المثال :

١ – أن أول قيمة للمتغير K ستكون 3 وآخر قيمة هي 7 .

٢ – أن تعليمة حلقة التنفيذ DO اقتصرت على المتغيرين i_I , i_F ولذا فأن الحاسب سيعتبر قيمة المتغير i_S هي 1 . وبالتالي فأن قيم K ستكون 7 ,6 ,5 ,4 ,3 على الترتيب .

مثـال ٦–١–٣ :

```
      IDRIS = 5
      MORAD = 6Ø
      KAMEL = 1Ø
      DO 7 KOUNT = IDRIS, MORAD, KAMEL
 7    ═══
```

وفي هذا المثال نجد أن :

١ – تعليمة حلقة التنفيذ المتكرر تحتوي على أسماء لمتغيرات صحيحة وليست قيماً صحيحة ، فالمتغير i في الصورة العامة والذي يمثل العداد في تلك التعليمة إستبدل بالمتغير KOUNT ، المتغير الصحيح i_I بالمتغير IDRIS ، والمتغير الصحيح i_F بالمتغير MORAD ، والمتغير الصحيح i_S بالمتغير KAMEL .

٢ – لابد من تعريف الحاسب مسبقاً بقيم المتغيرات الصحيحة KAMEL, MORAD, IDRIS مع مراعاة أن قيمة المتغير الصحيح i_F ستكون أكبر من أو تساوي قيمة المتغير الصحيح i_I وأن قيم المتغيرات الصحيحة الثلاثة لابد أن تكون موجبة .

ملاحظة :

ليس من الضروري أن تكون المتغيرات الصحيحة i_S, i_F, i_I جميعها قيماً صحيحة موجبة أو أسماء لمتغيرات صحيحة ولكن قد تحوي تعليمة حلقة التنفيذ المتكرر خليط من القيم والمتغيرات الصحيحة على سبيل المثال :

```
      ISTART = 10
      ISTEP = 5
      DO 9 L = ISTART,100,ISTEP
      ===
      ===
9     ===
```

مثال ٦-١-٤ :

أوجد قيمة M التي سيكتبها الحاسب من البرنامج التالي :

```
      M = -50
      DO 10 K = 5,20,5
10    M = M + K
      WRITE (7,15) M
15    FORMAT ('MM=', I3)
```

لاحظ إستخدام العداد K كمتغير صحيح في حساب قيم M في التعليمة رقم 10 .

– في البداية فأن قيمة M هي 50- .

– عندما يبدأ الحاسب في تنفيذ تعليمة حلقة التنفيذ المتكرر DO سيكوّن التنفيذ كالتالي :

التكرار رقم	قيمة K	قيمة M
1	5	− 50 + 5 = −45
2	10	− 45 + 10 = 35
3	15	− 35 + 15 = −20
4	20	− 20 + 20 = 0

– عند وصول قيمة المتغير K الى قيمته النهائية (2Ø في مثالنا هذا) ، يبدأ الحاسب في الخروج من حلقة التنفيذ المتكرر ليكمل تنفيذ بقية تعليمات البرنامـج أي كتابة قيمة M . وبذا سنتوقع من الحاسب أن يكتب النتيجة التالية : bM = bbØ

(**سؤال** : حاول كتابة البرنامـج السابق بدون إستخدام تعليمة حلقة التنفيذ المتكرر) .

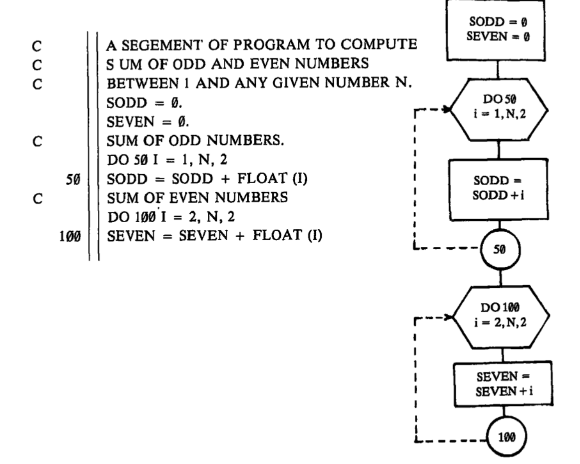
مثـال ٦-١-٥ :

اكتب برنامـج لحساب مجموع الأعداد الفردية وكذلك مجموع الأعداد الزوجية الواقعة بين العددين 1 , N .

لنفرض أن المتغير الذي سيحتوي على مجموع الأعداد الفردية هو SODD .
وأن المتغير الذي سيحتوي على مجموع الأعداد الزوجية هو SEVEN .

وحيث أنه سيتم فيهما عمليات تجميع ، فلابد أولاً من تصفيرهما ، أي جعل :

SEVEN = Ø. ; SODD = Ø.



```
C        A SEGEMENT OF PROGRAM TO COMPUTE
C        S UM OF ODD AND EVEN NUMBERS
C        BETWEEN 1 AND ANY GIVEN NUMBER N.
         SODD = Ø.
         SEVEN = Ø.
C        SUM OF ODD NUMBERS.
         DO 5Ø I = 1, N, 2
    5Ø   SODD = SODD + FLOAT (I)
C        SUM OF EVEN NUMBERS
         DO 1ØØ I = 2, N, 2
   1ØØ   SEVEN = SEVEN + FLOAT (I)
```

نلاحظ في هذا المثال :

١ – أن هناك حلقتي تنفيذ أحداهما لحساب SODD والأخرى لحساب SEVEN وفي الأثنتين كان المتغير الصحيح i (الذي يمثل العداد) واحدا لم يتغير . وفي هذا الاجراء نوع من توفير خلايا التخزين في الحاسب . ولن يكون هناك تداخل بين قيمة المتغير الصحيح i في حلقة التنفيذ الأولى وقيمته في حلقة التنفيذ الثانية ، اذ أنه بمجرد الانتهاء من تنفيذ الأولى ستكون قيمة i هي N أو (N-1) اذا كانت N زوجية . وعند البدء في تنفيذ حلقة التنفيذ الثانية سيأخذ المتغير الصحيح i القيمة الابتدائية i_I في تلك الحلقة وهي 2 ، ولن يستطيع الحاسب عمل ذلك الا بعد الانتهاء تماماً من تنفيذ حلقة التنفيذ الأولى .

٢ – أهمية المتغير الصحيح i_S وارتباطه بالمتغير الصحيح i_I ، حيث أمكن عن طريق هذا الارتباط الحصول على الأعداد الفردية والزوجية مباشرة .

٦–٢ تعليمة الاستمرار أو المواصلة CONTINUE Statement

وكما يتضح من إسم تلك التعليمة وهو الاستمرار أو المواصلة ، ولذا فأن الحاسب عندما يصل اليها في البرنامـج لتنفيذها فلن يكون عليه الا أن يستمر في تنفيذ تعليمات البرنامـج طبقاً لما يقوم به من حسابات ، ولذا فهي تختلف عن بقية التعليمات التي سبق شرحها والتي كان لكل منها هدف تعمله في البرنامج . ولذا أيضاً يطلق على مثل هذه الأنواع من التعليمات بالتعليمات الخادعة أو الزائفة Dummy Statements .

ومع ذلك ففي بعض المواضع من البرنامـج أحياناً ، لانستطيع الاستغناء عنها ، بل تصبح ضرورية الاستخدام كما سيتبين من الأمثلة التالية . وغالباً ما تكون هي التعليمة النهائية لحلقة تنفيذ متكررة ، خاصة اذا كانت تلك الحلقة التنفيذية تحوي بداخلها تعليمة اذا IF .

مثـال ٦–٢–١ :

```
      ===
10    X = 5.
      CONTINUE
15    X = X + 3.
      ===
```

في هذا المثال عندما يصل الحاسب الى تنفيذ التعليمة التي رقمها 10 فأنه سيخصص القيمة 5 للمتغير X . بعد ذلك ينزل الى التعليمة التالية والتي تبلغ الحاسب باستمرار التنفيذ ، وبالتالي فليس

أمام الحاسب سوى الاستمرار في تنفيذ تعليمات البرنامج ولذا يبدأ الحاسب في تنفيذ التعليمة التي رقمها 15 والتي تجعله يخصص القيمة 8 للمتغير X .

ومن الواضح أن وضع تعليمة الاستمرار في مثل تلك الظروف ليس لها داع لأنها لن تغير من حسابات البرنامج ، كما أنها لن تغير من سير تنفيذ تعليمات البرنامج ولذا يمكن الاستغناء عنها للمساعدة في جعل البرنامج قصيراً .

مثـال ٦-٢-٢ :

```
   |  X = Ø.
   |  DO 1Ø I = 1, 3
   |  X = X + 3.
1Ø |  CONTINUE
   |  Y = X * X
```

في هذا المثال نجد أن آخر تعليمة تنفيذية في حلقة التنفيذ المتكرر DO هي تعليمة الاستمرار ، وبالتالي فأن الحاسب في كل مرة يصل فيها الى تلك التعليمة فما عليه الا أن يغير قيمة I ويحسب قيمة X الجديدة . بعد تكرار تلك العملية ثلاث مرات يبدأ الحاسب في تنفيذ التعليمة الخاصة بحساب المتغير Y . ولذلك فأن المتغير X سيأخذ القيم 3 ,6 ,9 خلال تنفيذ حلقة التنفيذ المتكرر ، ثم بعد ذلك يتم حساب قيمة المتغير Y والذي سيساوي 81 .

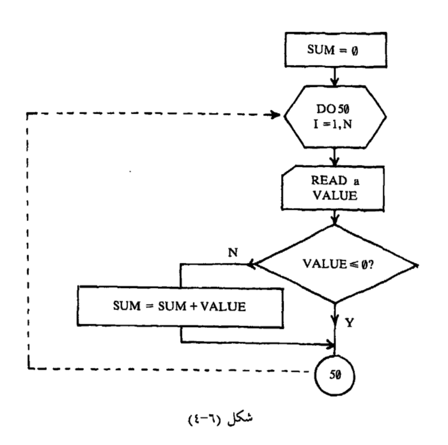
ولذلك فأن وضع تعليمة الاستمرار هنا لالزوم لها أيضاً ، لأننا لو كتبنا البرنامج بدونها فلن تتغير قيم X أو Y . ويكون البرنامج كالتالي :

```
   |  X = Ø.
   |  DO 1Ø I = 1,3
1Ø |  X = X + 3.
   |  Y = X * X
```

مثـال ٦-٢-٣ :

مجموعة من القيم عددها N بعضها موجب والآخر سالب ، وبعد أن يتم قراءة كل قيمة على حدة يتم جمع القيم الموجبة فقط منها ، إكتب برنامجاً لتحقيق ذلك .

لنفترض أن كل قيمة سيتم قراءتها عن طريق المتغير VALUE ، كما أن مجموع القيم الموجبة سيخزن في المتغير SUM ، لذا يمكن رسم مخطط التدفق كالتالي :



شكل (٦-٤)

```
C      A PROGRAM TO CALCULATE SUM OF + VE NUMBERS
       SUM = 0.
       DO 50 I = 1, N
       READ (5,5) VALUE
       IF (VALUE. LE. 0.) GO TO 50
       SUM = SUM + VALUE
50     CONTINUE
```

في هذه المشكلة تتضح أهمية تعليمة الاستمرار ، حيث أنه لايوجد بديل لها لتكون آخر تعليمة تنفيذية في حلقة التنفيذ المتكرر DO فبفضلها أمكننا التهرب من جيمع القيم السالبة في المتغير SUM والذي كان لابد سيحدث اذا لم توجد تعليمة الاستمرار في هذا الموضع ، وكانت التعليمة التي رقمها 50 ، أي آخر تعليمة في حلقة التنفيذ هي :

```
50     SUM = SUM + VALUE
```

ففي تلك الحالة فأن المتغير VALUE كان سيتم جمعه على المتغير SUM سواءً كان سالباً أم موجباً .

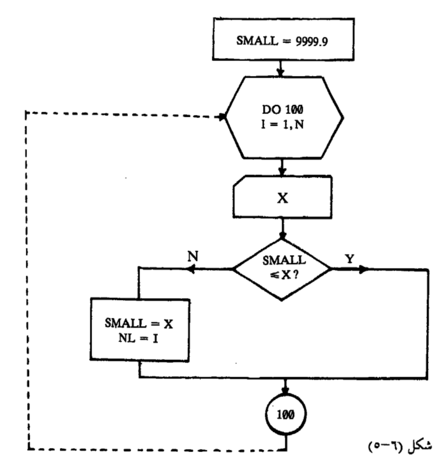
مثـال ٦-٢-٤ :

مجموعة من الأعداد الموجبة عددها N ، ادخلت الى الحاسب بطريقة عشوائية (أي ليست بطريقة تصاعدية للأعداد أو تنازلية) . إكتب برنامج لتعيين أصغر قيمة في هذه الأعداد وترتيبه في مجموعة تلك الأعداد .

لايجاد قيمة أصغر عدد في مجموعة أعداد ، سنبدأ بإفتراض أن أصغر عدد SMALL هو عدد له قيمة كبيرة جداً أو كبير نسبياً عن أي عدد في مجموعة الأعداد (كما يمكن أن تكون قيمة المتغير SMALL هو قيمة أول عدد في المجموعة) . ونبدأ في مقارنة المتغير SMALL بمجموعة الأعداد ، وكلما وجدنا عدداً أصغر من SMALL نحتفظ به ونخصص قيمته الى المتغير SMALL ونحتفظ بترتيبه في مجموعة الأعداد وهكذا .

ولايجاد أكبر عدد ، يمكن إتباع نفس الخطوات بإفتراض عدد صغير جداً في البداية ثم حجز قيمة وترتيب أي عدد له قيمة أكبر من قيمة ذلك العدد .

ويمكن تصور مخطط التدفق كالتالي :



شكل (٦-٥)

ويكون الجزء من البرنامج لحل تلك المشكلة كالتالي :

```
C        A SEGEMEENT OF PROGRAM TO FIND THE SMALLEST NUMBER.
         SMALL = 9999.9
         DO 100 I = 1, N
         READ (5 , 30) X
30       FORMAT (F 8.2)
         IF (SMALL. LE. X) GO TO 100
         SMALL = X ........................ SMALL حجز قيمة العدد الأصغر في المتغير
         NL = I ........................... NL حجز ترتيب العدد الأصغر في المتغير
100      CONTINUE
```

٦-٣ حلقات التنفيذ المتداخلة : Nested DO Loops

مقدمة :

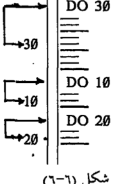
ماذا يقصد بحلقات التنفيذ المتداخلة ؟

علمنا فيما سبق معنى حلقة التنفيذ المتكررة وكيفية ومجالات إستخدامها وتبين لنا أن الحاسب عندما يبدأ في تنفيذ أي حلقة تنفيذ متكررة تصادفه في البرنامج ، فأنه يتوقف عند تلك الحلقة ويكرر تنفيذ التعليمات التي تحتويها عدداً من المرات يبدأ من i_I الى i_F ولاينتقل الى خارج حلقة التنفيذ الا تحت تأثير أحد إحتمالين :

١ - أن يصل المتغير i (الذي يمثل عداد حلقة التنفيذ) الى القيمة النهائية i_F ،

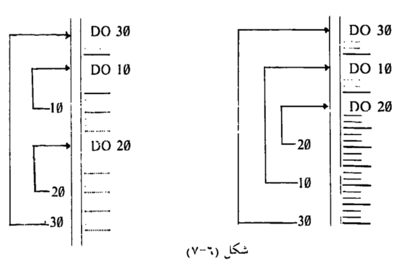
٢ - أن تحتوي تعليمات حلقة التنفيذ على إحدى تعليمات التحكم أو الانتقال والتي قد تتسبب في الخروج من تلك الحلقة .

وقد يحتوي البرنامج على اكثر من حلقة تنفيذ كل منها متباعدة عن الأخرى أو إحداها تتبع سابقتها مباشرة كما في شكل (٦-٦) وفي جميع الأحوال فأن الحاسب لن ينتقل الى تنفيذ حلقة منها الا بعد الانتهاء من تنفيذ سابقتها أو الخروج منها.



شكل (٦-٦)

وحلقات التنفيذ المتكررة المبينة بالشكل (٦-٦) تأتي في شكل متتال إحداها وراء الأخرى ، ولكن هل من الممكن أن تكون هناك اكثر من حلقة تنفيذ متكررة تأتي في شكل **متواز** ؟ - أي هل من الممكن أن تتخيل وجود الحلقات الثلاث السابقة مثلاً في إحدى الصور التالية في شكل (٦-٧) ؟



شكل (٦-٧)

والجواب على ذلك هو أن لغة الفورتران تقبل وجود مثل تلك الحلقات ولكن تحت شروط يجب مراعاتها وسنتحدث عنها فيما بعد ، ولكن مايجب أن نعرفه الآن أن وجود مجموعة حلقات تنفيذ متكررة كما في الشكلين السابقين يطلق عليها إسم حلقات التنفيذ المتداخلة .

والآن يتبقى السؤال :

كيف يقوم الحاسب بتنفيذ تلك المجموعة من حلقات التنفيذ المتكررة المتداخلة ؟

وللاجابة على ذلك السؤال ، سنحاول أولاً الاجابة على السؤال التالي :

في علم الجبر ، اذا أعطيت الكميتين الجبريتين التاليتين فكيف يمكنك حلهما ؟

[A + 3 (B − 2C) − 5 (3D + 2E)] , [A − 2{B + 3 (C − 2D) + 4} + 5E]

فنحن نعلم من قواعد الجبر البسيطة أنه اذا كانت هناك كميات جبرية كالمبينة فلحسابها يلزم أولاً فك أو انهاء الأقواس الداخلية ثم الانتقال بعد ذلك الى الأقواس الخارجية التي تليها وهكذا .

فلحساب الكمية الجبرية الأولى : [A + 3 (B − 2C) − 5 (3D + 2E)]

يجب أولاً حساب القوس الداخلي الأول : 3 (B − 2C) = 3B − 6C

ثم حساب القوس الداخلي الثاني : 5 (3D + 2E) = 15 D + 10 E

ثم تنتقل الى فك أو إنهاء القوس الخارجي ليصبح : A + 3B − 6C − 15D − 10E

ولحساب الكمية الجبرية الثانية : [A − 2 { B + 3 (c − 2D) + 4 } + 5E]

نبدأ أولاً بفك الأقواس الداخلية من النوع (   ) : 3 (C − 2D) = 3C − 6D

ثم ننتقل الى فك الأقواس الوسطى من النوع {   }

2 { B + 3C − 6D + 4 } = 2B + 6C − 12 D + 8

وأخيراً ننتقل الى فك أو إنهاء الأقواس الخارجية من النوع [   ] ، فيكون المقدار هو :

A − 2B − 6C + 12D − 8 + 5E

وهناك شبه كبير بين طريقة حساب مجموعة من حلقات التنفيذ المتداخلة وطريقة فك مجموعة من الأقواس المتداخلة ، ولذا فلتكن القاعدة العامة الآن هي أنه :

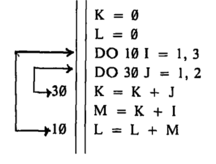
« اذا وجدت مجموعة من حلقات التنفيذ المتداخلة فأن الحاسب يقوم بإنهاء حلقات التنفيذ الداخلية ثم ينتقل الى التي خارجها الى أن يصل الى حلقة التنفيذ الخارجية » .

والفارق الوحيد بين الطريقتين هو أنه بمجرد فك الأقواس فأننا نستطيع بعد ذلك فك الأقواس التي تليها وهكذا . وفك كل نوع من الأقواس يتم مرة واحدة فقط ، بينما عند حساب مجموعة من حلقات التنفيذ المتداخلة فأنه كلما وصلت حلقة التنفيذ الداخلية الى نهايتها تبدأ حلقة التنفيذ التي خارجها في التغير خطوة واحدة وهكذا حتى تصل الحلقتين التنفيذيتين الى نهايتهما ، عندئذ تبدأ الحلقة التنفيذية التي تليهما في التغير خطوة ، وتستمر تلك العملية حتى تصل جميع الحلقات التنفيذية الى قيمة i_F في كل منها ثم يبدأ الحاسب بعد ذلك في تنفيذ التعليمات التي تلي تلك المجموعة من الحلقات التنفيذية المتداخلة .

وهذه العملية تشبه عداد الكيلومترات في السيارة والذي يتكون من مجموعة متداخلة من التروس ، الداخلي منها يعد الآحاد والذي يتلوه للعشرات ثم المئات وهكذا . وكلما وصل ترس الآحاد الى الرقم 9 يبدأ ترس العشرات في الانتقال خطوة ليبدأ ترس الآحاد مرة أخرى في الدوران حتى الى الرقم 9 ، وهكذا حتى يصل العداد بجميع تروسه الى قيمته العظمى ولتكن 99999 .

مثـال ٦-٣-١ :

أوجد قيم K , Lمن البرنامج التالي :



```
      K = 0
      L = 0
      DO 10 I = 1, 3
      DO 30 J = 1, 2
30    K = K + J
      M = K + I
10    L = L + M
```

- عند تنفيذ الحاسب لهذا الجزء من البرنامج فأنه سيضع قيمتى L , K تساوي أصفاراً .
- لفك حلقتى التنفيذ المتكررتين المتداخلتين يبدأ الحاسب بأخذ أول قيمة من المتغير I . أي يضع I = 1 ثم ينهي حساب حلقة التنفيذ الداخلية فيأخذ J = 1 ثم يحسب قيمة K ، ثم يأخذ قيمة J = 2 ليحسب قيمة K

- بوصول قيمة J الى قيمتها النهائية يبدأ الحاسب في تنفيذ التعليمات التي بعد ذلك حتى يصل الى آخر تعليمة في حلقة التنفيذ الخارجية والتي رقمها 10 .
- بوصول الحاسب الى التعليمة رقم 10 ، يبدأ في تغيير قيمة I لتكون I = 2 ، ثم يبدأ في فلك حلقة التنفيذ الداخلية . أي يأخذ J = 1 ثم J = 2 وهكذا حتى تصل قيمة I الى 3 وقيمة J الى 2 .

وبالتالي فأن خطوات الحساب ستكون كالتالي :

```
K  =  0
L  =  0
I = 1 :
        J = 1 → K = 0 + 1 = 1
        J = 2 → K = 1 + 2 = 3
                M = 3 + 1 = 4
                L = 0 + 4 = 4
I = 2 :
        J = 1 → K = 3 + 1 =  4
        J = 2 → K = 4 + 2 =  6
                M = 6 + 2 =  8
                L = 4 + 8 = 12
I = 3 :
        J = 1 → K = 6 + 1 =  7
        J = 2 → K = 7 + 2 =  9
                M = 9 + 3 = 12
                L = 12 + 12 = 24
```

أي أنه في نهاية تنفيذ حلقتى التنفيذ المتداخليين ستكون قيمة M هي 12 ، قيمة L هي 24 .

مثـال ٦-٣-٢ :

في مجموعة الأعداد من 1 الى 9 ، اكتب برنامـج لايجاد مجموع كل ثلاث أعداد متتالية منها ، أي يراد إيجاد (1 + 2 + 3) , (4 + 5 + 6) , (7 + 8 + 9)

```
      DO 10 I = 1,7,3
      ISUM = 0
      I1 = I
      I2 = I + 2
      DO 20 J = I1, I2
20    ISUM = ISUM + J
10    WRITE (6,15) ISUM
15    FORMAT (I5)
```

٦-٤ ملحوظات على إستخدام حلقة التنفيذ المتكررة : DO

ملحوظة ١ :

١ – كما سبق أن أشرنا أنه يجب :

(أ) أن تكون جميع المتغيرات الصحيحة i_I , i_F , i_S إما قيماً صحيحة موجبة أو متغيرات صحيحة سبق اعطاؤها قيماً صحيحة موجبة في البرنامج أو خليط من القيم الصحيحة الموجبة والمتغيرات الصحيحة .

(ب) أن تكون قيمة i_I أقل من أو تساوي i_F ، وإذا حدث أن كانت قيمة i_I اكبر من أو تساوي قيمة i_F ، فأن الحاسب سيقوم بتنفيذ حلقة التنفيذ المتكرر مرة واحدة فقط مستخدماً قيمة العداد $i = i_I$.

مثـال :

```
      DO 3 L = 10 , 5
      ___
      ___
3     ___
```

ستقوم الحاسب في هذه التعليمة بأخذ قيمة L تساوي 10 ، وينفذ بقية التعليمات التي تحويها حلقة التنفيذ المتكرر مرة واحدة فقط . أي يبدأ بعد ذلك مباشرة في تنفيذ التعليمات التي تلي التعليمة التي رقمها 3 :

ملحوظة ٢ : لايجوز تغيير قيمة المتغير i داخل التنفيذ المتكرر DO .

مثـال :

```
      DO 10 K = 1, 100, 2
      ......
      K = I * J/2
      ......
10    ......
```

والسبب في ذلك أن المتغير الصحيح K في حلقة التنفيذ المتكرر DO تعمل كعداد لمعرفة عدد المرات التي تم فيها تنفيذ التعليمات التي تحويها الحلقة ، وبالتالي فلا يجوز عمل إضطراب لذلك المتغير أثناء تنفيذ تعليمات حلقة التنفيذ المتكرر . ولكن يمكن إستخدام المتغير الصحيح في إحدى أو بعض التعليمات التي تحويها حلقة التنفيذ المتكرر كما في مثال ٦-٣-٢ .

كما لايجوز أيضاً تغيير قيم المتغيرات الصحيحة i_I أو i_F أو i_S داخل حلقة التنفيذ المتكرر .

مثـال :

```
      J = 50
      DO 7 K = 5,J
      ===
      J = K + 2
      ===
7     ===
```

ملحوظـة ٣ :

لايجوز أن تكون المتغيرات الصحيحة i_I ، i_F ، i_S مصحوبة بعمليات حسابية كالجمع أو الطرح ... الخ .

مثـال :

```
      DO 9 M = J + 2, K, 3
      ===
9     ===
```

فالتعليمة بهذه الصورة غير صالحة وعرضة لظهور أحد أنواع الأخطاء التي يكتشفها الحاسب . ولكن يمكن بتحوير بسيط إجراء نفس العملية بإفتراض متغير آخر جديد J2 مثلاً ، يحل محل العملية الحسابية J + 2 ، وبالتالي تتحول التعليمة الى :

```
      J2 = J + 2
      DO 9 M = J2, K, 3
      ===
9     ===
```

ملحوظـة ٤ :

يجب أن تكون آخر تعليمة في حلقة التنفيذ المتكررة هي تعليمة تنفيذية مثل تعليمات القراءة والكتابة وتعليمات الاستمرار ، **ولا يجب** أن تكون إحدى تعليمات التحكم أو الانتقال أو تعليمات صيغ الادخال أو الاخراج (FORMAT) . فعلى سبيل المثال فأن التعليمات التالية **غير صحيحة** .

```
      DO 7 K = 1, N
      ===
7     IF (          )
```

```
      DO 5 I = 1, L
      ===
5     GO TO 10
      ===
10    ===
```

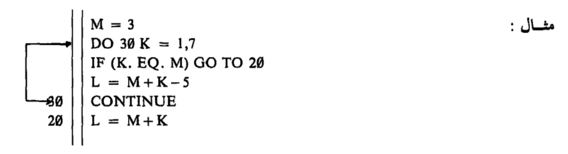
```
      DO 10 J = 1, K
      READ (7, 10) X
10    FORMAT (F 8.2)
```

وقد يكون ذلك هو السبب في أن كثيراً من المبرمجين يفضلون إنهاء أي تعليمة حلقة تنفيذ متكرر بتعليمة الاستمرارCONTINUE سواءكان لها أهمية في وضعها أم لا .

ملحوظة ٥ :

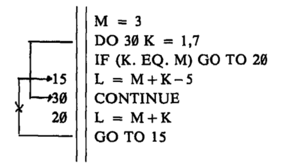
يمكن الخروج من حلقة التنفيذ المتكرر **ولكن** لايمكن الدخول فيها .

مثـال :



```
      M = 3
      DO 30 K = 1,7
      IF (K. EQ. M) GO TO 20
      L = M+K-5
30    CONTINUE
20    L = M+K
```

في هذا المثال عندما تصل قيمة K الى 3 فأن الحاسب سيتوقف عن تنفيذ بقية تكرارات الحلقة وينتقل الى التعليمة التي رقمها 20 والتي توجد خارج نطاق حلقة التنفيذ المتكررة . ولكن لايمكن كتابة التعليمات كالتالي مثلاً :



```
      M = 3
      DO 30 K = 1,7
      IF (K. EQ. M) GO TO 20
15    L = M+K-5
30    CONTINUE
20    L = M+K
      GO TO 15
```

حيث أنه **لايجوز** الدخول في حلقة التنفيذ المتكررة عند موضع أي تعليمة منها سواءً كان الدخول من تعليمة تسبق حلقة التنفيذ المتكررة أو من تعليمة تأتي بعد حلقة التنفيذ المتكررة كما هو مبين في الجزء السابق من البرنامج .

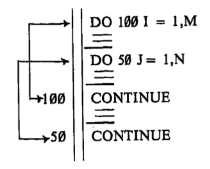
ملحوظة ٦ :

داخل حلقة التنفيذ المتكررة **يمكن** الانتقال من تعليمة الى تعليمة أخرى سابقة لها أو لاحقة . فعلى سبيل المثال فأن الجزء التالي من البرنامج مسموح به ، حيث أن عملية الانتقال تتم داخل حلقة التنفيذ نفسها .

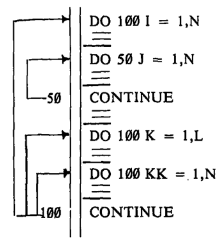
```
          M = 0
          DO 100 J = 1, N
   20     IF (I - J) 50, 100, 60
   50     M = M + J - I
          I = I + 1
          GO TO 20
   60     M = M + I - J
  100     CONTINUE
```

ملحوظة ٧ :

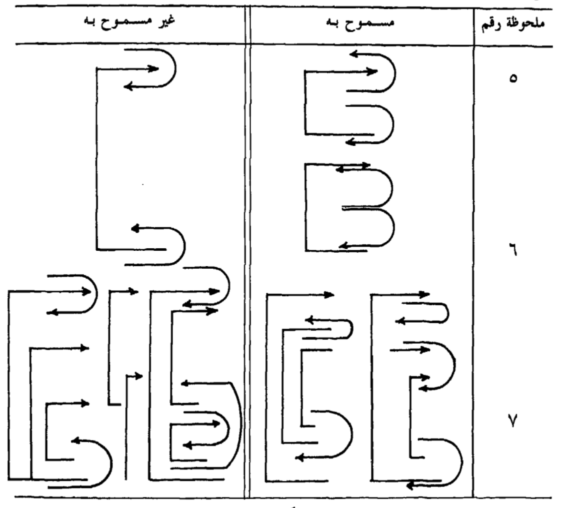
في حلقات التنفيذ المتداخلة يجب أن تقع حلقات التنفيذ الداخلية **بأكملها** داخل حلقات التنفيذ الخارجية **ولايجوز** أن تتقاطع حلقتى تنفيذ . فعلى سبيل المثال فأن الجزء التالي من البرنامج **غير** مسموح به حيث أن هناك تقاطع بين حلقتى تنفيذ متكرر .



كما يجوز أن تشترك اكثر من حلقة تنفيذ متكرر متداخلة في تعليمة نهاية واحدة . فعلى سبيل المثال ، فأن جزء البرنامج التالي مسموح به حيث لايوجد تقاطع بين حلقات التنفيذ بينما تشترك اكثر من حلقة تنفيذ متكررة في نهاية واحدة .



وفي النهاية بأفتراض أن العلامة   تمثل حلقة تنفيذ متكررة وأن العلامة
تمثل تعليمة انتقال ، فاننا يمكن تلخيص الملاحظات ٥ ، ٦ ، ٧ بالأشكال التالية :



شكل (٦-٨)

٦-٥ : تمارين على إستخدام حلقات التنفيذ المتكرر DO loops

١ - في تعليمات حلقات التنفيذ المتكررة التالية ، أوجد عدد مرات التكرار والقيمة النهائية التي سيأخذها المتغير الصحيح الذي يمثل عداد الحلقة .

a) DO 5Ø K = 3,16,4

b) DO 1Ø L = 2,14,3

c) DO 2Ø KOUNT = 2,9,4

d) DO 15 M = 75,82,1Ø

٢ - أوجد الأخطاء - إن وجدت - في التعليمات التالية :

a) DO 2Ø L1ØØ = L2ØØ, L3ØØ, L4ØØ

b) DO 9 K = 2, 7 * M, 3

c) DO 1Ø, J = 5,2Ø,K,

d) DO 5 L = I,J,L,

٣ - أوجد قيمة L بعد تنفيذ أجزاء البرامج التالية :

a)

```
     I=2
     J=3
     DO 1Ø K = 3,14,I
1Ø   L = J+K
     L = 2*L
```

b)

```
     K=5
     L=Ø
     DO 15 I = 1,9,2
15   L = L+K+I
```

c)

```
     L=2
     I=3
     DO 8 K = 1,5
     L = L+K+I
     IF (L. GT. 22) GO TO 5
8    CONTINUE
5    L = L+5
```

d)

```
     L=19
     M=2
     DO 15 J = 2,9,M
     IF (J. GT. 6) GO TO 15
     L = L-2*J
15   CONTINUE
     L = M*L
```

e)

```
      L = 19
      M = 2
10    DO 15 J = 2,9,M
      IF (J. EQ. 6) GO TO 10
      L = L - 2*J
15    CONTINUE
      L = M*L
```

٤ – أوجد الأخطاء في أجزاء البرامج التالية :

a)

```
10    X = A+B
      DO 50 L = 1,9,2
15    IF (X.LT.FLOAT (L)) GO TO 50
      GO TO 10
50    CONTINUE
      X = X - B
      GO TO 15
```

b)

```
       ═══
10     X = A+B
       ═══
       DO 5Ø L = 1,9,2
       ═══
       IF (X.LT.FLOAT (L)) GO TO 1Ø
       ═══
15     X = X - B
       ═══
       DO 1ØØ K = 1,7
       ═══
       GO TO 15
       ═══
5Ø     CONTINUE
1ØØ    CONTINUE
```

c)

```
       ═══
1Ø     X = A+B
       GO TO 3Ø
15     DO 2Ø L = 1,7
       ═══
       X = X - FLOAT (L)
       ═══
5Ø     Y = X*X
       ═══
4Ø     POLYN = Y+X-3.
       ═══
       DO 2Ø K = 2,9
       ═══
       IF (POLYN. GT 7.) GO TO 4Ø
       ═══
3Ø     POLYN  = POLYN - 3.
       ═══
2Ø     CONTINUE
       ═══
       GO TO 5Ø
       ═══
```

d)

```
      ===
      GO TO 30
      ===
      DO 10 K = 1,7
      ===
15    L = K+2
      ===
      DO 20 M = 1,6
      ===
      IF (M. EQ. 3) GO TO 30
20    CONTINUE
30    M = M+L
35    DO 50 L = 1,5
      ===
      GO TO 40
      ===
50    CONTINUE
      ===
      GO TO 35
40    K = K+L
      ===
10    CONTINUE
      ===
```

٥ – اكتب جزء من برنامج يكتب الرقم 20 عشرون مرة ، الرقم 19 تسعة عشرة مرة وهكذا .

٦ – اكتب جزء من برنامج يكتب الأعداد الفردية الواقعة بين العددين N , 1 وبجانب كل عدد يكتب مربعة ومكعبة .

٧ – بافتراض عددين موجبين K , N حيث N اكبر من K . اكتب جزء من برنامج باستخدام حلقة التنفيذ المتكرر لكتابة العدد N ثم العدد (N – 1) ، (N – 2) وهكذا حتى العدد K .

٨ – بافتراض أن قيم المتغيرات A , B , N معطاه ، اكتب جزء من برنامج لحساب :

$$S = \frac{1}{A} + \frac{1}{A+B} + \frac{1}{A+2B} + \ldots + \frac{1}{A+NB}$$

٩ – بافتراض أن قيمة المتغير N معروفة ، اكتب برنامج لحساب :

$$Result = \frac{1}{1^2} \cdot \frac{3}{2^2} \cdot \frac{5}{3^2} \ldots \frac{2N-1}{N^2}$$

١٠– اكتب برنامج يعطي جميع الأعداد الفردية الصحيحة الموجبة والتي أقل من 100 ، دون إعطاء تلك الأعداد الفردية الموجبة التي تقبل القسمة على 7 .

١١- الأعداد الثلاثة الموجبة الصحيحة A,B,C حيث A أقل من B , B أقل من C تكون مثلثا قائم الزاوية اذا كان $A^2+B^2 = C^2$ ، اكتب برنامج لايجاد جميع قيم C,B,A التي تكون مثلثات قائمة الزوايا حيث B , A أقل من 25 .

١٢- بافتراض مجموعة من البطاقات عددها N ، على كل منها رقم صحيح موجب . اكتب برنامج لايجاد عدد الأرقام الزوجية وكذلك عدد الأرقام الفردية .

١٣- بافتراض المعادلة : $Y = 2X^3 - 5X^3 + 3X + 2$

اكتب برنامج يحسب قيمة Y التي تعتمد على المتغير X عندما :

(أ) $7 \leq X \leq 7-$ بزيادة قيمة X في كل مرة بمقدار 0.1

(ب) $I \leq X \leq J$ بزيادة قيمة X في كل مرة بمقدار $\frac{1}{N}$ حيث المتغيرات N,J,I تكون قد سبق قراءتها وفيها المتغير I أقل من المتغير J .

١٤- بافتراض أن هناك ٢٥ رقما صحيحا موجبا يتم ادخالها الى الحاسب عن طريق البطاقات . اكتب الجزء من البرنامج الذي يعطي ثاني اكبر تلك الأعداد .

١٥- بافتراض المعادلة : $Z = X^2 - 2XY + 3Y^2 - 8X + 3Y - 8$

اكتب برنامج يحسب قيمة Z عند قيم متغيرة للمتغيرين Y , X تتغير من 3- الى 3 وكل منها يزيد بمقدار 0.2 .

١٦- في التمرين السابق ، أوجد اكبر قيمة يأخذها المتغير Z عند قيم Y,X المختلفة .

١٧- يمكن حساب قيمة النسبة التقريبية π باستخدام كثيرة الحدود :

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots$$

اكتب برنامج لحساب :

(أ) مجموع المائة حد الأولى من كثيرة الحدود .

(ب) مجموع المائة حد الأولى من كثيرة الحدود بترتيب عكسي .

(جـ) أوجد الفرق بين النتيجتين .

١٨- مجموعة من الأعداد عددها N يتم ادخال احداها وراء الآخر . اكتب برنامج لحساب اكبر تلك الأعداد وأصغرها .

١٩- بافتراض مجموعة من البطاقات على كل منها رقم صحيح موجب . اكتب برنامج لايجاد اكبر عدد زوجي في هذه المجموعة . أو يكتب العنوان التالي :

«NO EVEN INTEGER» في حالة ما اذا كانت جميع الأعداد فردية .

٢٠- بافتراض مجموعة من القراءات عددها N تمثل ظاهرة معينة لمتغير X . اكتب برنامج يحسب من هذه القراءات الانحراف المعياري لها STDV . استخدم القانون :

$$STDV = \sqrt{\frac{\sum x_i^2}{N} - \left(\frac{\sum x_i}{N}\right)^2}$$

٢١- يتم تحويل الدرجة X بالنظام « الستينى » الى الدرجة XR بالنظام « الدائري » بأستخدام القانون :

$$XR = X. \pi/180$$

حيث π هي النسبة التقريبية والتي تساوي 3.141592654

- اكتب برنامج لحساب XR لقيم من X تتغير من ٩٠ الى ١٨٠ وكل درجة تزيد عن سابقتها بمقدار نصف درجة .
- استخراج النتائج على الصورة التالية :

ƀƀƀƀƀDEGREESƀCONVERSIONƀ TABLE

X	XR
xxx	xx.xxxxx

« تمـــارين »

١ - أوجد الخطأ في كل تعليمة من تعليمات حلقة التنفيذ المتكرر التالية (إن وجد) :

```
a )     DO  10, I = 1,N
b )     DO  15, FROM I = 30 TO N-2
c )     DO  10 I = 1, N, L-1
d )     DO  20 J = 12, 62, I
e )     DO  8 MEAN = MABLE, ABLE, NAGLE
f )     DO  7 I = 3, I, 3
```

٢ - في أجزاء البرامج التالية ، أوجد الأخطاء (إن وجدت) :

a)

```
      DO 3 I = 1, 6
      DO 7 K = 1, I
      DO 3 J = 1, K
  7   K = K+1
  3   M = I*J*K
```

b)

```
      :
 10   L = L=1
      DO 20 L = 1, 5
 15   IF (L. EQ. 6) GO TO 10
 20   CONTINUE
      L = L+1
      GO TO 15
      :
```

c)

```
        :
        GO TO 25
        DO 10 I = 1, 3
15      II = II + I
        DO 20 J = 1, 3
        IF (J. EQ. II) GO TO 25
20      CONTINUE
25      JJ = JJ + J
        DO 30 K = 1, 6
        IF (JJ. EQ. K) GO TO 35
30      CONTINUE
35      MULT = I*JJ*K - 10
        IF (MULT. GT. 0) GO TO 25
10      CONTINUE
        GO TO 15
```

٣ – باستخدام تعليمة التنفيذ المتكرر ، اكتب برنامج لحساب :

$$SUM = \frac{1}{1(A+B)} + \frac{1}{2(A+2B)} + \frac{1}{3(A+3B)} + \dots + \frac{1}{N(A+NB)}$$

٤ – اكتب برنامج لحساب الدرجات المئوية T_c بدلالة الدرجات الفهرنهيتية T_F باستخدام العلاقة :

$$T_c \equiv \frac{5}{9} (T_F - 32)$$

حيث T_F تأخذ القيم بين ٣٢ و ٢١٢ بزيادة قدرها $\frac{١}{٢}$

٥ – يمكن حساب النسبة التقريبية π من العلاقة :

$$\pi^2 = 6 \quad \left(1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \dots\right)$$

اكتب برنامج لحساب π واحسب عدد الحدود N التي يجب جمعها بحيث أن : $(\pi^2(N) - \pi^2(N-1))$ تكون أقل من أو تساوي 0.000001

٦ – بافتراض أن : $R = A^3 + 2AB - 5B^2 + 6A - 8B - 3$

اكتب برنامج لحساب قيمة R لقيم مختلفة من B,A تتغير من 5- الى 5+ بزيادة قدرها 0.1 .

٧ – اكتب برنامج لحساب مضروب أي عدد صحيح موجب N ، حيث مضروب N (N!) تعطي من :

$$N! = N.(N-1) \bullet (N-2) \ldots 3.2.1$$

٨ – صيغة ستيرلنج التقريبية لحساب مضروب أي عدد صحيح موجب N هي :

$$N! \cong (2\pi N)^{\frac{1}{2}}, \quad (\frac{N}{e})^{N}$$

حيث : π هي النسبة التقريبية وتساوي تقريباً 3.141592654

e يتم حسابها عن طريق القانون :

$$e \cong 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots \cong 2.7182818$$

اكتب برنامج لحساب مضروب أعداد بين 1 و 1Ø

٩ – يمكن حساب جيب أي زاوية مقدرة بالتقدير الدائري من القانون :

$$\text{Sin } X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \ldots$$

اكتب برنامج لحساب جيب زوايا بين Ø.1 و 5.Ø بزيادة قدرها Ø.3

١٠– لحساب الجذر التربيعي لرقم حقيقي موجب T فأن إحدى الطرق لتحقيق ذلك هي طريقة نيوتن والتي تفترض قيمة تقريبية أولية X للجذر ، ثم يتم حساب قيمة EST أقرب الى الجذر حيث :

$$EST = \emptyset.5\ (X + \frac{T}{X})$$

ثم يتم ابدال القيمة X بالقيمة المحسوبة EST وهكذا حتى يتحقق الشرط :

$$\frac{X - EST}{X} \leq \pm E$$

حيث E قيمة صغيرة جداً يتوقف إختيارها على درجة التقريب المطلوبة .

١١– إحدى الطرق لحساب الجذر النوني (N) لأي عدد صحيح موجب T هو إفتراض قيمة تقريبية أولية للجذر ولتكن X ، ثم حساب قيمة أقرب للجذر EST بإستخدام القانون :

$$EST = \frac{1}{N}\left((N-1)X + \frac{T}{X^{N-1}}\right)$$

ثم يتم ابدال القيمة X بالقيمة المحسوبة EST وهكذا حتى يكون الفرق بين قيمتين متتاليتين للمتغير EST أصغر من قيمة صغيرة جداً E ، يتوقف إختيارها على درجة التقريب المطلوبة .

اكتب برنامج لتحقيق ذلك .

الفصل السابع

المصفوفات

الفصل السابع
المصفوفات ARRAYS

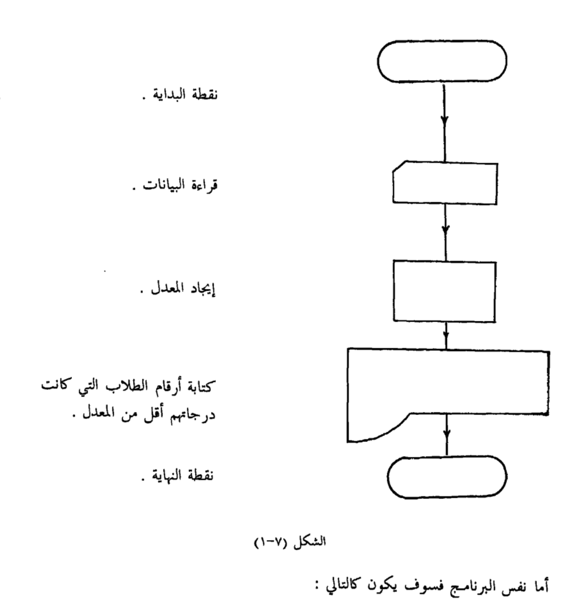
مقدمة :

تعلمنا الى الآن قدراً جيداً من لغة الفورتران بما يكفي لتنفيذ أي مقدار من العمليات الحسابية المعقدة ، ولكن هناك مجموعة خاصة من المسائل التي تحتاج الى تعليمات جديدة لتنفيذها رغم أن القدر الذي تعلمناه وحتى الآن يمكن استعماله في حل مثل هذه المسائل ولكن ذلك سيؤدي الى كتابة برامج ضخمة وطويلة ومملة مما يؤدي الى فقدان عامل السرعة واختصار الوقت والجهد والمال التي يتميز بها الحاسب الآلي . وبصفة عامة فأن هذه المسائل تتكون غالباً من تطبيقات عملية حيث تحتاج فيها الى إستعمال بعض البيانات اكثر من مرة ، فبدلاً من إعطاء كل رقم في البيان اسم خاص به نقوم باعطاء جميع عناصر البيان نفس الاسم وذلك باستخدام مايسمى بالمصفوفات ARRAYS .

٧-١ : أهمية المصفوفات :

لننظر الى المثال التالي لندرك أهمية إستعمال المصفوفات في لغة الفورتران . لنفرض أن أحد الفصول الدارسية المكون من عشرة طلاب قد قدم أحد الامتحانات العامة في مادة ما ولنفترض أن لدينا بيان يتكون من أرقام الطلاب ودرجاتهم التي حصلوا عليها في هذا الامتحان ، ولنفترض أن البيان مكون من عشرة اسطر بحيث يحتوي كل سطر على رقم الطالب والدرجة التي حصل عليها والآن لنحاول أن نكتب برنامجاً بلغة الفورتران يوجد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام في هذا الامتحان .

في البداية نقوم برسم مخطط التدفق للبرنامج .



الشكل (٧-١)

أما نفس البرنامـج فسوف يكون كالتالي :

```
COMMENT ---- PROGRAM TO LIST THE STUDENTS NUMBERS WITH
C        BELOW AVERAGE GRADE IN A MATH. TEST.
         INTEGER SN1,SN2,SN3, SN4, SN5,SN6,SN7,SN8,SN9,SN1Ø
         REAL G1,G2,G3,G4,G5,G6,G7,G8,G9,G1Ø
         READ (5,2Ø) SN1, G1
         READ (5,2Ø) SN2, G2
         READ (5,2Ø) SN3, G3
         READ (5,2Ø) SN4, G4
         READ (5,2Ø) SN5, G5
```

```
      READ (5,20) SN6, G6
      READ (5,20) SN7, G7
      READ (5,20) SN8, G8
      READ (5,20) SN9, G9
      READ (5,20) SN10, G10
   20 FORMAT (I6, F5. 1)
C
      AVE = (G1 + G2 + G3 + G4 + G5 + G6 + G7 + G8 + G9 + G10)/10.0
C
      WRITE (6,30)
   30 FORMAT ('1 STUDENTS WITH BELOW AVERAGE GRADES'/)
C
      IF (G1. LE. AVE) WRITE (6, 40) SN1
      IF (G2. LE. AVE) WRITE (6, 40) SN2
      IF (G3. LE. AVE) WRITE (6, 40) SN3
      IF (G4. LE. AVE) WRITE (6, 40) SN4
      IF (G5. LE. AVE) WRITE (6, 40) SN5
      IF (G6. LE. AVE) WRITE (6, 40) SN6
      IF (G7. LE. AVE) WRITE (6, 40) SN7
      IF (G8. LE. AVE) WRITE (6, 40) SN8
      IF (G9. LE. AVE) WRITE (6, 40) SN9
      IF (G10. LE. AVE) WRITE (6, 40) SN10
   40 FORMAT (' ',I6)
C
      STOP
      END
```

data

```
816439 b 72.5
813951 b 85.0
804613 b 69.3
825601 b 90.0
804501 b 58.5
819456 b 77.0
826351 b 80.0
819678 b 62.5
806351 b 74.0
813532 b 66.0
```

output

```
STUDENTS WITH BELOW AVERAGE GRADES
816439
804613
804501
819678
813532
```

الشكل (٧-٢)

في الواقع أنه اذا ماتفحصنا البرنامج تفحصاً دقيقاً فإنه يتضح لنا أن هذه هي أفضل طريقة لكتابة البرنامج على الوجه المطلوب مستعملين في ذلك ماتعلمناه في الفصول السابقة ، ولكن مهلاً فلربما يقول قائل أنه كان بإمكاننا إستعمال الحلقة التكرارية DO loop وبالتالي إختصار عدد كبير من الأسطر ، أي بدلاً من كتابة عشرة اسطر لجملة READ فأنه بإمكاننا إستعمال المتغيرين SN و G لقراءة القيم العشرة الموجودة في البيان على الوجه التالي :

```
      INTEGER SN
      SUM = 0.0
      DO 25 I = 1, 10
         READ (5, 20) SN, G
         SUM = SUM + G
20    FORMAT (I6, F5.1)
25    CONTINUE
      AVE = SUM/10.0
```

الشكل (٧-٣)

وفي الواقع فأن هذا ممكن اذا كان المطلوب هو ايجاد المعدل فقط ولكن كيف يمكننا ايجاد أرقام الطلبة الذين حصلوا على درجات أقل من المعدل في حين أن خليتى SN و G تحويان على رقمين فقط هما الموجودين في نهاية البيان أي رقم الطالب العاشر ودرجته في الامتحان بينما الأرقام الأولى قد أزيلت من الذاكرة كما تعلمنا سابقا .

اذا فالبرنامج المكتوب في الشكل (٧-٣) هو أفضل مايمكن كتابته حالياً على ضوء ماتعلمناه حتى نهاية الفصل السابق . وقد يبدو للقارىء أن طول البرنامج مناسب ومعقول وبالتالي فلا داعي لتعلم اداه جديدة لتبسيط البرنامج اكثر من هذا ، ولكن نفترض أن لدينا مائة طالب أو خمسمائة طالب قد تقدموا لهذا الامتحان ، وعندئذ يتحتم علينا اذا ما أردنا إستعمال هذه الطريقة أن نكتب

٥٠٠ جملة لقراءة البيان و ٥٠٠ جملة شرطية IF Statement لايجاد المطلوب وبالتالي فأن المهمة تكون اضاعة للوقت والجهد وموضعاً للسخرية والتهكم .

وللتغلب على هذه المشكلة فاننا نحتاج الى طريقة لتغيير الخلية المستعملة بواسطة جملة أقرأ بدلاً من تغيير الجملة نفسها ، وبالفعل نستطيع أن نفعل هذا بواسطة استعمال المصفوفات ARRAYS .

٧-٢ المصفوفات :

المصفوفة AN ARRAY عبارة عن مجموعة من خلايا الذاكرة التي تحمل نفس الاسم ويمكن التفريق بين عناصر هذه المجموعة بواسطة استعمال دليل عددي index أو subscript يكون مرتبطاً بالاسم المستعمل في البرنامج بحيث يكتب الاسم ثم يكتب على يمينه الدليل العددي بين قوسين كما هو موضح في الشكل (٧-٤ ب) .

وفي المثال السابق الذي استعرضناه مؤخراً نجد أننا نحتاج في الواقع الى مصفوفتين يتكون كل منها من عشرة عناصر ، وعندئذ بدلاً من أن نستعمل عشرة أسماء مختلفة SN1 و SN2 و...و SN1Ø يمكننا أن نستعمل مصفوفة واحدة تحمل الاسم SN وتتكون من العناصر SN (1) و SN (2) و .. وهكذا الى SN (1Ø)

(أ)		(ب) SN	
SN 1			SN (1)
SN 2			SN (2)
SN 3			SN (3)
SN 4			SN (4)
SN 5			SN (5)
SN 6			SN (6)
SN 7			SN (7)
SN 8			SN (8)
SN 9			SN (9)
SN 1Ø			SN (1Ø)

الشكل (٧-٤)

لاحظ أن (أ) يمثل عشرة خلايا تحمل عشرة أسماء مختلفة بينما (ب) يمثل مصفوفة تحمل نفس الاسم SN وتتكون من عشرة خلايا مختلفة تتميز باختلاف الدليل العددي لكل خلية من هذه الخلايا العشر . وفي الواقع لو أن الخاصية الوحيدة لاستعمال المصفوفة هي كتابة (5) SN بدلاً من SN 5 أو (8) SN بدلاً من SN 8 لما استفدنا شيئاً جديداً على الاطلاق ولكن الخاصية الهامة للمصفوفات تكمن في إمكانية كتابة الدليل العددي في صورة مقدار جبري من الممكن أن تتغير قيمته العددية من وقت لآخر خلال تنفيذ البرنامج فمثلاً :

```
K = 1
READ (5,500) SN (K), G (K)
```

تعني أن الحاسب سيقرأ قيمتى (1) SN و (1) G لأن قيمة K تساوي 1 ولو كانت قيمة K مساوية 4 لقرأ الحاسب قيمتى (4) SN و (4) G . أما لو كانت قيمة K تساوي 1 فأن جملة :

READ (5, 500) SN (4 * K + 1), G (K + 2)

تعني أن الحاسب سيقرأ قيمتى (5) SN و (3) G وهكذا ، ولا بد لنا هنا أن نلفت الانتباه أن الدليل العددي يجب أن يكون عدداً صحيحاً كما يجب أن يكون موجباً في معظم الأنظمة المتبعة .

٧-٣ الجملة المبينة للمصفوفة : ARRAY declarator

لكي نستعمل مصفوفة في برنامج فأننا نحتاج الى توضيح ذلك للحاسب في بداية البرنامج كما فعلنا سابقاً مع المتغيرات الحقيقية والصحيحة . والجملة المبينة للمصفوفة تقوم بثلاث مهام رئيسية هي :

١ - تحدد اسم المصفوفة .
٢ - تحدد طول المصفوفة .
٣ - تحدد نوعية المصفوفة من حيث أنها من نوع الأعداد الصحيحة فقط أو الأعداد الحقيقية فقط علماً بأنه لايمكن الجمع بين نوعين مختلفين من الأرقام في نفس المصفوفة ، أي أنه لايمكن للعنصر SN 3 أن يكون عدد صحيحاً INTEGER بينما SN 6 يمثل عدداً حقيقياً REAL .

الجملة المبينة للمصفوفة	Array declarator
الشكل العام	Type NAME (LEN)

أما Type فهي إما أن تكون REAL و INTEGER و NAME اسم للمصفوفة لايتجاوز الست رموز تتكون من أحرف وأرقام وتبدأ بحرف ولا يحتوي الاسم على علامات خاصة .

و LEN عدد صحيح ثابت اكبر من الصفر .

المعنى : اشعار الحاسب بأن NAME هو اسم لمصفوفة تتكون من عدد من خلايا الذاكرة عددها يساوي LEN ومحتواها هو من نوع Type .

أمثلـة :

```
INTEGER A, BX (16) , STUDNT (34)
REAL NADIA (5) , Z3 (14) , XY
```

هاتان الجملتان لو وجدتا في برنامـج واحد فأنها تأمر الحاسب بتشكيل أربعة مصفوفات هي : BX, STUDNT, NADIA, Z3 حسب الأطوال المعطاه بين القوسين وحسب نوع المتغير المبين بكلمتى INTEGER أو REAL فمثلاً STUDNT اسم لمصفوفة طولها ٣٤ أي أنها تتكون من ٣٤ خلية ذات محتوى من نوع الأعداد الصحيحة .

تنبيـه :

لابد لنا هنا من الاشارة الى الفارق بين طول المصفوفة LEN وبين المتغير الذي ترتيبه LEN في المصفوفة فمثلاً بالنسبة للمثال الذي تكلمنا عنه في بداية هذا الفصل كان لدينا عشرة أرقام للطلاب رمزنالها بـ SN ولهذا فأننا نحتاج الجملة المبينة للمصفوفة SN .

```
INTEGER SN (10)
```

وكما هو واضح من الشكل (٧-٤ ب) فأن SN (10) بحد ذاتها تعني العنصر العاشر في المصفوفة بينما SN (10) الموجودة في الجملة أعلاه تعني أن طول المصفوفة هو ١٠ أي أنها تتكون من عشر عناصر .

٧-٤ عناصر المصفوفة :

عنصر المصفوفة والذي يشكل خلية ذاكرة في المصفوفة يستعمل تماماً كما تستعمل خلايا الذاكرة الأخرى من حيث وجودها في جمل أخرى من جمل الفورتران كالجمل الشرطية والجبرية واقرأ واكتب ... الخ ، الا أن هناك اضافة بسيطة بالنسبة لكتابة العنصر نفسه تتلخص في لزوم اضافة الدليل العددي الخاص بذلك العنصر لاسم المصفوفة كما بينا سابقاً . وللعلم فأن العناصر تبدأ دائماً من ١ الى الحد الأعلى الذي يمثل طول المصفوفة ولهذا فأن آخر عنصر في المصفوفة له دليل عددي يساوي طول المصفوفة .

أمثلـة : هذه بعض الأمثلة التي توضح كيفية استعمال عنصر ما في المصفوفة .

```
A (5) = 47.0 * B + C1
READ (6, 506) BX (3 * J - 4)
IF (T(7) + A(3) .LT. T (4) * 3.0) STOP
Y (M) = Y (M - 1) + SUM
X (8) = A(8) * Y(3)
```

أما الآن وباستعمال المصفوفات فانه بامكاننا إعادة كتابة البرنامج السابق لايجاد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام بصورة اكثر بساطة ووضوحا .

```
COMMENT --- PROGRAM TO LIST STUDENT'S NUMBERS  WITH BELOW
C            AVERAGE GRADE IN A MATH. TEST.
             INTEGER SN (10)
             REAL G (10), AVE
             INTEGER I
C
C    **      STORE DATA
             I = 1
     10      READ (5,20) SN(I), G(I)
     20      FORMAT (I6,F5.1)
             I = I + 1
             IF (I .LE. 10) GO TO 10
C
             AVE =(G(1) + G(2) + G(3) + G(4) + G(6) + G(7) + G(8) + G(9) + G(10))
          1  /10.0
```

```
C
          WRITE (6,30)
    30    FORMAT ('1', STUDENTS WITH BELOW AVERAGE GRADES'/)
C
          I = 1
    35    IF (G(I) .LE. AVE) WRITE (6,40) SN (I)
    40    FORMAT ('      ', I6)
          I = I + 1
          IF (I .LE. 10) GO TO 35
          STOP
          END
```

الشكل (٧-٥)

ومما لاشك فيه أننا اذا أعطينا الحاسب نفس البيان data الموجود في نهاية البرنامج السابق فاننا بالتأكيد سنحصل على نفس النتيجة output الموجودة في الشكل (٧-٢) .

وعلى الطالب أن يلاحظ الفارق الكبير بين البرنامجين من حيث الوضوح واليسر وتوفير الوقت سواءاً في قراءة البرنامج أو كتابته .

٧-٥ جملة تحديد البعد : DIMENSION Statement

تستعمل هذه الجملة لتحديد اسم وطول المصفوفة فقط دون تحديد لنوعية المصفوفة من حيث كونها ذات قيم عشرية أو صحيحة . أما الشكل العام لها فهو :

```
DIMENSION NAME (LEN)
```

حيث NAME و LEN قد عرفت سابقاً في شرح الجملة المبينة للمصفوفة . وتفتقد هذه الجملة لميزة هامة هي تحديد نوعية المصفوفة منذ النظرة الأولى كما هو الحال مع الجملة المبينة للمصفوفة التي تكلمنا عنها في الصفحات القليلة الماضية .

مثـال :

لنفترض أن لدينا مصفوفة اسمها MILE وطولها 20 وذات قيم عشرية وأردنا إستعمالها ضمن أحد البرنامج ، ففي هذه الحالة يمكننا أن نوصل هذه التعليمات الى الحاسب بطريقتين مختلفتين هما كالتالي :

```
DIMENSION MILE (20)
REAL MILE
```

أو

```
REAL MILE (20)
```

ومن هنا يتضح تفوق الطريقة الثانية على الطريقة الأولى من حيث قصدها ومدلولها المساوي للطريقة الأولى التي إستعملنا فيها سطرين بدلاً من السطر الواحد . لاحظ أنه في الطريقة الأولى لم نضف طول المصفوفة الى الاسم ، أي أنه لايجوز لنا أن نكتب :

```
DIMENSION MILE (20)
REAL MILE (20)
```

لأن هذا يعني بالنسبة للحاسب أن هناك مصفوفتين مختلفتين تحملان نفس الاسم مما يؤدي الى إرسال إشعار بحدوث خطأ في البرنامج .

مثال : لو أفترض أن كتبنا الجملة DIMENSION MILE (20)
دون أي إضافة أخرى لتحديد نوعية MILE فأن الجملة صحيحة ولكن طبقاً لقاعدة التعيين التلقائي التي تكلمنا عنها في الفصل الثاني فأن MILE مصفوفة طولها20 وتحمل قيماً **صحيحة** .

« تمـارين عـامـة »

التمرين الأول :

تحت أي الشروط يكون العنصر X(I) والعنصر X(J) يعنيان نفس العنصر في المصفوفة X ؟

التمرين الثاني :

ما الذي سيطبعه الحاسب بعد تنفيذ البرنامج التالي ؟

```
      INTEGER A (10) , I
      A (1) = 0
      A (2) = 1
      I = 3
10    A (I) = A (I-1) + A (I-2)
      I = I + 1
      IF (I .LE. 10) GO TO 10
      WRITE (6,20) A(1), A(2), A(3), A(4), A(5)
      WRITE (6,20) A(6), A(7), A(8), A(9), A(10)
20    FORMAT (5I4)
      STOP
      END
```

التمرين الثالث :

لنفترض أن B عبارة عن مصفوفة من عشر عناصر وأن I و J أسماء متغيرات صحيحة وأن قيمة I تساوي 3 بينما قيمة J تساوي 7 . أي من الجمل التالية يعتبر صحيحاً ؟ واذا لم تكن جملة ما صحيحة .. وضح لماذا ؟

```
B(3) = B(I)
B(I) = B(I-1)
B(J) = B(2 * I-1)
B(4) = B(J+1) * B(I * J - 21)
B(2*I) = B(J+5)
B(1.7) = 0.0
```

التمرين الرابع :

أي من جمل التعيين التالية صحيح ؟

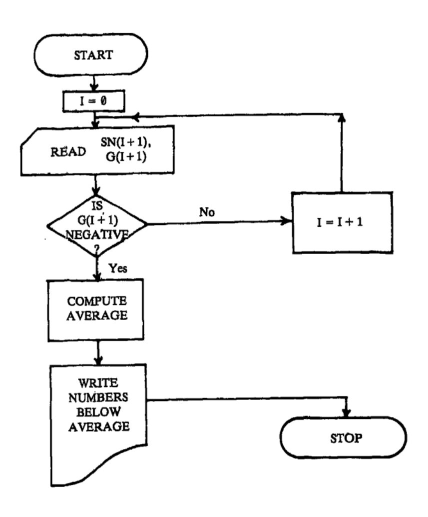
```
REAL A(10)
INTEGER A(13 - 2)
INTEGER A(I)
REAL X(150) , YFL (B520)
REAL ALI (15.0)
INTEGER SAUDIA (25) XT (3)
```

حل أفضـــل :

لنفترض أن لدينا ثلاثون طالباً في الفصل الدراسي الذي تكلمنا عنه سابقاً ولنفترض أنهم أعطوا امتحاناً في مادة ما ولنفترض أننا نريد أن نكتب برنامجاً لايجاد أرقام الطلاب الذين حصلوا على درجات أقل من المعدل العام للفصل . في الواقع أن البرنامـج الموضح في الشكل (٧–٥) يفي بالغرض تماماً اذا ما أستبدلنا الرقم 10 بالرقم 30 ولكن على شرط أن يتقدم جميع طلاب الفصل الى الامتحان لأنه في حالة تغيب أحدهم فأن حساب المعدل يختلف كما أن البرنامـج لايصبح صالحاً لايجاد المطلوب . لذا كان لابد من وجود طريقة لمعرفة عدد الطلبة الذين تقدموا للامتحان . بحيث تكون هذه الطريقة صالحة لأي عدد من الطلبة أقل أو يساوي العدد الكلي للطلبة وهو ثلاثون . أما أفضل الطرق الحالية لايجاد العدد المتقدم للامتحان فهي معرفة عدد الأرقام التي تقرأ من البيان المعطى للحاسب والذي يشتمل على أرقام ودرجات الطلاب المشاركين في الامتحان . ولهذا فأننا سنضيف في نهاية البيان رقماً سالباً مثلاً يمثل درجة طالب وعندئذ يدرك الحاسب بواسطة إستعمال جملة IF الشرطية بأن البيان الفعلي قد أنتهى فعلاً وأن هذا الرقم السالب الذي قرىء أخيراً يعتبر إشعاراً للحاسب بانتهاء البيان الفعلي .

الشكل (٧–٦) يمثل مخططاً تدفقياً لبرنامجنا المعدل وهو في الواقع يشبه الى حد كبير مخطط التدفق المعملي في الشكل (٧–١) مع الاختلاف الوحيد الممثل في تتبع الحاسب للعدد الفعلي للطلبة المتقدمين للامتحان .

ومما تجدر الاشارة اليه هنا أن برنامجاً كهذا يعتبر من البرامج الأساسية التي تتكرر مثيلاتها في المستقبل لدرجة كبيرة . لذا كان من الجدير بالطالب أن يفهم ويدرك النقاط الرئيسية في برنامـج كهذا حتى يسهل عليه الكثير في المستقبل .



شكل (٧-٦)

من الملاحظ في هذا المخطط التدفقي أن قيمة I النهائية ستكون مساوية للعدد الفعلي للطلبة المشاركين في الامتحان لأنه حالما يقرأ الحاسب قيمة سالبة لعلامة الطالب فأنه يتجه الى الخطوة التالية دون أن يغير من قيمة I النهائية ، وعلى الطالب أن يتأكد من صحة هذا المنطق بتجربة المخطط يدوياً لأربعة أو خمسة طلاب ليتأكد بنفسه من صحة الخطوات المتبعة في المخطط .

أما الآن فانه بإمكاننا بسهولة أن نكتب البرنامج بعد أن فهمنا مخطط التدفق للعمليات الحسابية والمنطقية التي سيقوم بها البرنامج لتنفيذ الدور المطلوب منه .

```
COMMENT --- PROGRAM TO LIST THE STUDENT S NUMBERS WITH
C          BELOW AVERAGE GRADE IN A MATH TEST.
           INTEGER SN (3Ø),I,K
           REAL G(3Ø) , AVE
C
C   **     STORE DATA
           I = Ø
    1Ø     READ (5,2Ø) SN(I+1) , G(I+1)
    2Ø     FORMAT (I6 , F5. 1)
           IF (G(I+1) .LT. Ø.Ø) GO TO 3Ø
           I = I+1
           GO TO 1Ø
C
C   **     COMPUTE THE AVERAGE
    3Ø     SUM = Ø.Ø
           AI = I
           K = 1
    4Ø     SUM = SUM + G(K)
           K = K+1
           IF (K .LE. I) GO TO 4Ø
           AVE = SUM / AI
C
           WRITE (6,5Ø)
    5Ø     FORMAT('1 STUDENTS WITH BELOW AVERAGE GRADES'/)
C
           K = 1
    6Ø     IF (G(K) .LE. AVE) WRITE (6,)Ø) SN(K)
    7Ø     FORMAT ('   ', I6)
           K = K+1
           IF (K .LE. I) GO TO 6Ø
           STOP
           END
```

الشكل (٧-٧)

٧-٦ المصفوفات وحلقات التنفيذ : ARRAYS AND DO LOOPS

في الواقع من أهم إستعمالات برامج الفورتران هي إستخدام المصفوفات كجزء من حلقات الـ DO فهما في الواقع مكملين لبعضهما لأن إستعمالهما معاً يدل بوضوح على كفاءة الحاسب الآلي ومقدرته العظيمة على إختزال الوقت وتوفير الجهد سواءً من ناحية كتابة البرامج أو من ناحية الوقت الذي يستغرقه تنفيذ البرنامج غالباً .

أما الآن فسنكتب نفس البرنامج السابق بإستعمال حلقات الـ DO وسنلاحظ أن البرنامج أصبح اكثر بساطة ووضوحاً .

```
COMMENT ---- PROGRAM TO LIST THE STUDENT'S NUMBERS WITH
C         BELOW AVERGE GRADE IN A MATH TEST
          INTEGER SN (30) , I , K , N
          REAL G (30) , AVE
          SUM = 0.0
C **      STORE DATA AND FIND SUM OF ACTUAL GRADES
          DO 10 I = 1,30
              READ (5,20) SN (I), G(I)
              IF (G(I) .LE. 0.0) GO TO 30
              SUM = SUM + G(I)
    10    CONTINUE
    20    FORMAT (I6, F5.1)
          N = I - 1
C **      COMPUTE THE AVERAGE
    30    AN = N
          AVE = SUM / AN
C
          WRITE (6,50)
    50    FORMAT ('1 STUDENTS WITH BELOW  AVERAGE GRADES'/)
C
          DO 60 K = 1 , N
              IF (G(K) .LE. AVE) WRITE (4,70) SN (K)
    60    CONTINUE
    70    FORMAT ('   ', I6)
          STOP
          END
```

الشكل (٧–٨)

٧–٧ بعض الأخطاء الشائعة عن المصفوفات :

عادة ما تكون المصفوفات مصدر إرباك للمبرمج المبتدىء . ولذلك فأن القائمة التالية ستساعد القارىء على تلافي كثير من الأخطاء الشائعة عن إستعمال المصفوفات

أولاً : ليست هناك علاقة بين قيمة الدليل العددي وبين العنصر نفسه فمثلاً ليست هناك علاقة بين العنصر SN(5) والرقم 5 الذي يمثل الدليل العددي للعنصر .

ثانياً : عندما يستعمل أحد المتغيرات كدليل عددي فأن الأهمية تكمن في قيمة المتغير وليس في إسم المتغير . فمثلاً قد تستعمل في إحدى مراحل البرنامج المتغير A(I) بينما حينا آخر قد تستعمل المتغير A(J) على الرغم من أننا نتعامل مع نفس المصفوفة . ومن الجائز أن يكون A(I) و A(J) يرمزان الى نفس العنصر تماماً اذا كانت قيمتي I و J متساويتان في إحدى مراحل البرنامج .

ثالثاً : قد يستعمل نفس المتغير كدليل عددي لمصفوفتين مختلفتين ، فمثلاً يجوز لنا أن نستعمل A (I) و B (I) في نفس الوقت كما فعلنا في البرنامج السابق عند إستعمال SN (I) و G (I) . مرة أخرى قيمة الدليل أهم من إسمه .

رابعاً : لاتستعمل المصفوفات اذا كنت في غنى عنها ، لأن ذلك قد يؤدي الى كتابة برامج غير مجدية إقتصادياً .

خامساً : القاعدة العامة في لغة الفورتران أن الحاسب يستطيع أن يتعامل مع عنصر واحد في وقت واحد ، أي أنه لايستطيع أن يتعامل مع اكثر من عنصر في مصفوفة في وقت واحد — فمثلاً في البرنامج السابق لو وضعنا الجملة التالية :

```
IF (G .LE. AVE) WRITE (6,3Ø) SN
```

لاتعني أن الحاسب سوف يختبر جميع عناصر المصفوفة G وفي الواقع أن هذه الجملة غير صحيحة في لغة الفورتران ولذلك لزم أن نقارن كل عنصر على حده كما فعلنا في البرنامج المكتوب في الشكل (٧–٨) .

سادساً : تذكر أنه لايمكن تغيير طول المصفوفة أثناء البرنامج ، ولذلك إذا لم تكن متأكداً من طول مصفوفة ما ، فما عليك الا أن تستعمل اكبر طول يمكن أن تصل اليه تلك المصفوفة كما فعلنا بالنسبة للفصل الذي يمكن أن يحتوي على ٣٠ طالب كحد أعلى انظر الشكل (٧–٨) .

٧–٨ المصفوفة ذات الدليلين العددين : Two-Dimensional Arrays

في بعض الأحيان يكون من المفيد أن يستعمل مصفوفة ذات طولين ولها دليلين عدديين قد يكون لهما نفس الطول أو قد يكونا مختلفين . ومثال ذلك A(5,3) وهنا ندرك أن A هو اسم لمصفوفة على شكل جدول به خمسة صفوف RAWS وثلاثة أعمدة COLUMNS . انظر الشكل (٧–٩) .

المصفوفة A

A (1,1)	A (1,2)	A (1,3)
A (2,1)	A (2,2)	A (2,3)
A (3,1)	A (3,2)	A (3,3)
A (4,1)	A (4,2)	A (4,3)
A (5,1)	A (5,2)	A (5,3)

عدد الصفوف 5

عدد الأعمدة = ٣

الشكل (٧-٩)

ومن الملاحظ أن الدليل العددي الأول يمثّل عدد الصفوف بينما الدليل العددي الثاني عدد الأعمدة . ومما تجدر الاشارة اليه هنا أن الحاسب يقوم بترتيب الخلايا بشكل عمودي كما هو موضح في الشكل (٧-١٠) .

العمود الأول	A (1,1)
	A (2,1)
	A (3,1)
	A (4,1)
	A (5,1)
العمود الثاني	A (1,2)
	⋮
	A (5,2)
العمود الثالث	A (1,3)
	⋮
	A (5,3)

الشكل (٧-١٠)

أي أن العنصر الذي ترتيبه ٧ مثلاً هو العنصر (2,2) A بينما (4,3) A يحمل الترتيب رقم ١٤ ، والعنصر (5,3) A ترتيبه ١٥ . أما عدد العناصر التي تتكون منها المصفوفة فهي تساوي حاصل ضرب عدد الصفوف في عدد الأعمدة ، فمثلاً المصفوفة (5,3) A تتكون من ١٥ عنصرا . ولا يفوتنا هنا أن نؤكد مرة أخرى على أن العنصر الأخير في المصفوفة له دليلان عدديان مساويان في الواقع للرقمين الذين يمثلان طول المصفوفة بينما ليست بين الطرفين أي علاقة على الاطلاق .

أمثلـة :

١ – `REAL X(3,2) , Y(4) , Z`

٢ – `INTEGER NUMBER (4,2) , C(10)`

٣ – `X(2,2) = X(1,2) * Y(3) + 9.0`

٤ – `C(5) = A(3) * C(4) / Z`

٥ – `WRITE (6,46) X(1,1) , X(2,1) , X(1,2)`

٦ – `READ (5,24) NUMBER (2,1) , C(8)`

وفي الواقع فأن هناك كثير من التطبيقات العملية التي يظهر فيها مدى الحاجة الى إستعمال هذا النوع من المصفوفات . وهنا نترك لأستاذ المادة مهمة إختيار المثال المناسب لطلبة الفصل الذين يدرسون المادة معه .

٧–٩ ادخال واخراج بيانات المصفوفات :

في هذه المرحلة من الكتاب يعتبر القارىء ملمـاً بقدر كاف من المعرفة بطرق كتابة المصفوفات ، ومن الملاحظ أنه حتى الآن عند استعمال جملة اكتب WRITE فأن المبرمـج عليه أن يعرف تماماً عند كتابة البرنامـج عدد عناصر المصفوفة التي يرغب في كتابتها بواسطة الحاسب . ومما لاشك فيه أن القدر الحالي من المعلومات يمكننا من كتابة عناصر أي مصفوفة باستعمال الحلقات التكرارية DO Statements ولكن هذا غالباً مايؤدي الى كتابة عنصر واحد في كل سطر من البيانات المتحصل عليها . أما اذا رغبنا في أن نكتب جميع عناصر المصفوفة دفعة واحدة في نفس السطر أو في اكثر من سطر اذا لم يتسع سطر واحد لجميع العناصر ، فأن الأمر يحتاج الى جهد اكبر .

ولتسهيل الأمر فأن هناك طريقة أخرى من طرق ادخال واخراج البيانات تستعمل بصفة خاصة في حالات كهذه مرتبطة بكتابة قيم المصفوفات تسمى الحلقات المباشرة Implied Do List .

وتتلخص هذه الطريقة في كتابة قائمة أسماء المتغيرات (التي غالباً ماتكون عناصر مصفوفة) متبوعة بالأدلة العددية التي تحدد عدد مرات تكرار كتابة أسماء المتغيرات في القائمة .

مثـال :

لنفترض أننا نريد أن نكتب قيم عناصر المصفوفة A من (1) A الى (N) A حيث N اسم لمتغير ذو قيمة عددية صحيحة INTEGER ، فبدلاً من كتابة الحلقة التكرارية .

```
      I = 1
30    WRITE (5,40) A (I)
      I = I + 1
      IF (I .LE. N) GO TO 30
```

نستطيع أن نستغني عن هذا كله بالجملة :

```
      WRITE (5,40) (A(I) , I = 1,N)
```

ومن الوضح أن من أهم مزايا استعمال جملة كهذه أنها أبسط كتابة وأسهل قراءة ، ومع ذلك فأن لها مزية هامة أخرى وهي أنها توجه الحاسب الى كتابة جميع قيم المصفوفة في سطر واحد (ان اتسع السطر لذلك) وعند عدم اتساع السطر لجميع القيم فأنه يقوم تلقائياً بكتابة بقية القيم في السطر التالي أو السطور التالية حسب حجم المصفوفة . أما في حالة استعمال الحلقات التنفيذية فأن الأمر يختلف حيث أن الحاسب يقوم بكتابة كل قيمة في سطر لأنه قام الحاسب بتنفيذ جملة اكتب فانه يبدأ بسطر جديد .

وفي الواقع فأن الحلقة المباشرة Implied Do List لايشترط فيها أن تكون مشابهة تماماً للجملة التي كتبناها في الفقرة الماضية ، فقائمة المتغيرات يمكن أن تحتوي على اكثر من متغير ، كما أن الدليل العددي يمكن أن يبدأ بأي عدد صحيح موجب اكبر من الواحد كما لايشترط أن تكون الزيادة بمقدار 1 في كل مرة . فأذاً يمكن للدليل العددي أن يبدأ بأي عدد صحيح موجب positive INTEGER كما يمكن أن يزيد كل مرة بأي عدد صحيح موجب . كما يمكن لنا أن نستعمل أسماء متغيرات لتقوم مقام الثوابت التي تحدد بداية الدليل العددي ونهايته والزيادة في كل مرة .

مثـال :

الجملة التالية تعتبر جملة صحيحة :

```
WRITE (5,270) (L , A (L) , B (L) , L = M, N, K)
```

فهي تأمر الحاسب بأن يعطي L قيمة مبدئية تساوي M ثم يكتب قيمة كل من L ,(L) A و (L) B ثم يزيد قيمة L بمقدار K ثم يكتب قيم L ,(L) A ,(L) B . وهكذا يستمر التكرار طالما أن قيمة L بعد كل زيادة لم تتجاوز قيمة N المعروفة سلفاً لدى الحاسب .

« تمـريـن »

إفرض في المثال السابق أن قيمة M تساوي 2 وقيمة N تساوي 10 وقيمة K تساوي 3 ، اكتب جزءاً من برنامـج يقوم بنفس مهمة الجملة أعلاه بدون استعمال الحلقات المباشرة .؟

اذا يمكن تلخيص الشكل العام للحلقة المباشرة كالتالي :-

الحلقة المباشرة Implied Do List

الشكل العـام READ / WRITE (List , V = s, b,i)

حيث أن List عبارة عن أي قائمة صحيحة .

V عبارة عن اسم الدليل العددي المتغير و s, b, i أعداد صحيحة ثابتة INTEGER Constants أو اسماء متغيرات صحيحة ليست عنصراً في مصفوفة وذات قيم موجبة .

المعنى :

يوجه الحاسب الى قراءة أو كتابة قائمة من المتغيرات المتكررة لكل قيمة من V بدءاً بقيمة s وتزاد بمقدار i طالما أن V لاتتجاوز قيمة b .

ملاحظـة : عندما تكون قيمة i مساوية 1 فانه بالامكان تجاهلها وعدم كتابتها .

امثلـــة :

```
READ (5,1000) (XT(I) , I=1,15,3)
WRITE (6,48) (A(K-2) , K=3,N)
WRITE (6,600) (NUM, V(NUM) , FX , NUM=L,20,INC)
```

انفاذ المصفوفات : **Array Transmission**

المقصود هنا أنه بمجرد كتابة اسم المصفوفة فقط في قائمة جملة اقرأ / اكتب فأن كامل المصفوفة يمكن ارسالها الى الطابع لكتابتها أو يمكن تلقيها بواسطة الجهاز القارىء لقراءتها .

مثـــال :

```
REAL A, B, X(3)
READ (5,3Ø1) A,X,B
```

هاتين الجملتين صحيحتان ومساويتان تماماً للجملتين التاليتين :

```
REAL A, B, X(3)
READ (5,3Ø1) A, X(1), X(2), X(3), B
```

٧-١٠ تداخل الطرق المتكررة المباشرة :

بالامكان كما هو واضح في الشكل العام للحلقة المباشرة أن يسمح بتداخل اكثر من حلقة مباشرة . أي أنه بالامكان اكمال حلقة مباشرة داخلية تماماً عند كل تكرار للحلقة المباشرة الخارجية . وللتوضيح فاننا نضع الأمثلة التالية التي سوف توضح كثيراً مما نعني :

مثـــال (١) :

الجملة : `WRITE (6,1ØØ) C, ((B(J), (A(1), I = 1,3), J = 1,2),D`

تعتبر مساوية تماماً للجملة :

```
WRITE (6,1ØØ) C, B(1), A(1), A(2), A(3), B(2), A(1), A(2), A(3),D
```

مثـــال (٢) :

الجملـــة : `WRITE (6,2ØØ) ((A(I,J) , J = 1,3) , I = 1,2)`

تعتبر مساوية تماماً للجملة :

```
WRITE (6,2ØØ) A(1,1) , A(1,2) , A(1,3) , A(2,1) , A(2,2) , A(2,3)
```

وعموماً فأنه يمكن ادخال أو طباعة عناصر أية مصفوفة بطرق عديدة تختلف فيما بينها تبعاً لعاملين :

١ - طرق تنفيذ تعليمات القراءة أو الطباعة .

٢ - الجملة الشارحة المرافقة لتلك التعليمات .

وفيما يلي سنستعرض بعض الأمثلة ، بعرض طرق ادخال أو طباعة عناصر مصفوفات ذات بعد واحد (أي ذات دليل عددي واحد) وذات بعدين Two dimensional array . وما ينطبق على تعليمات القراءة أو الادخال ، ينطبق أيضاً على تعليمات الطباعة .

أولاً – في حالة المصفوفات ذات البعد الواحد :

في هذه الحالة فأن عناصر المصفوفة المراد ادخالها الى الحاسب أو طباعتها تكون مرتبة في شكل صف أو عمود . وشكل المصفوفة في هذه الحالة غير ذي أهمية ولكن الأهم هو وضع كل عنصر في مكانه المخصص له في المصفوفة ، والذي يتحكم في ذلك هو قيمة الدليل العددي كما تبين من الأمثلة السابقة .

مثـال (١) :

لنفترض أن هناك ٥٠٠ قيمة يراد قراءتها وتخزينها في مصفوفة ذات بعد واحد X . هناك طريقتين لادخال تلك القيم في الحاسب :

١ – اما بادخال تلك القيم قيمة بعد الأخرى (أي قيمة في كل سطر) ، ويتم ذلك بطريقتين :

```
C      METHOD A
       DIMENSION X (500)
       DO 8 K = 1, 500
   8   READ (5,9) X (K)
   9   FORMAT (10 F 8.2)
```

وفي هذه الحالة لابد من أعطاء **كل قيمة على حده** ، حتى مع وجود الجملة الشارحة التي تسمح بأخذ عشرة قيم متتالية في كل مرة ، حيث أن حلقة التنفيذ ستجبر الحاسب في أن يقرأ قيمة واحدة فقط كل مرة . ولذا فعلينا أن ندخل الخمسمائة قيمة في خمسمائة سطر متتالية .

```
C      METHOD B
       DIMENSION X (500)
       READ (5,11) (X(K), K = 1,500)
  11   FORMAT (F8.2)
```

تعتبر هذه الحالة عكسية للحالة السابقة ، فبينما الجملة الشارحة في الطريقة السابقة تسمح باعطاء الحاسب عشرة قيم في كل مرة ولكن تعليمة القراءة هي التي حددت قيمة واحدة وليس أكثر ، فأنه في هذه الحالة فأن تعليمة القراءة تسمح باعطاء الحاسب أكثر من قيمة في كل مرة ، بينما الجملة الشارحة لتلك التعليمة ستجبر الحاسب في أن يقرأ قيمة واحدة فقط .

٢ – أو بادخال تلك القيم في مجموعات (أي كل مجموعة من القيم في سطر واحد) ، ويتم ذلك بطريقة واحدة فقط كالتالي :

```
        DIMENSION X (500)
        READ (5,9) (X(K), K = 1,500)
9       FORMAT (10F8.2)
```

في هذه الحالة فأن تعليمة القراءة تسمح للحاسب بقراءة قيم المصفوفة كلها في سطر واحد (إن امكن ذلك) ، ولكن الجملة الشارحة ستجبر الحاسب في هذه الحالة في أن يقرأ عشرة قيم تتبعها العشرة قيم التالية وهكذا . ولذا فيجب علينا في هذه الحالة أن نعطي الحاسب كل **عشرة قيم** فقط في سطر حيث أنه لن يشعر الحاسب بالقيم التي تعطى له بعد القيمة العاشرة في نفس السطر ، كما أنه في حالة ما اذا كان عدد القيم في السطر أقل من عشرة ، فأنه سيعتبر أن بقية القيم تساوي صفرا .

ومن الحالات المختلفة السابقة تتضح العلاقة التي يجب أخذها في الاعتبار بين تعليمة القراءة والجملة الشارحة لتلك التعليمة .

ملحوظة هامة :

في تعليمات القراءة أو الادخال يجب الا يزيد عدد الحروف والارقام التي يراد ادخالها عن ٨٠ في كل سطر ، بينما في تعليمات الطباعة فان عدد الحروف والأرقام التي يمكن طباعتها أو اخراجها **يتوقف** على وحدة الاخراج أو الطباعة ففي الشاشات أو البطاقات لايجب أن يزيد عن ٨٠ حرفا ورقما ، بينما في الطابعات يمكن أن يصل الى ١٢٠ أو ١٣٢ . ولذا – في المثال السابق – نظراً لأن كل قيمة تأخذ الصورة F 8.2 ، كان لزاماً علينا الا يزيد عدد القيم التي يمكن ادخالها الى الحاسب عن عشرة قيم في كل سطر . وبناءاً عليه فلو أن كل قيمة كانت في الصورة F 5.2 مثلاً ، لأمكن أن نعطي الحاسب كل ١٦ قيمة في سطر واحد .

ثانياً – في حالة المصفوفات ذات البعدين :

في هذه الحالة فان كل قيمة يراد ادخالها في المصفوفة أو استخراجها منها ، يتحدد موضعها عن طريق الدليلين العدديين الذي يحدد أولهما رقم الصف الذي سيتم ادخال القيمة فيه وثانيهما يحدد رقم العمود . ولقراءة أو طباعة مصفوفة ذات بعدين ، فان هناك اكثر من طريقة سنحاول التفرقة بينها بافتراض المثال التالي :

مثال (٢) :

لنفترض أن هناك ٥٠٠ قيمة ، يراد قراءتها وتخزينها في مصفوفة X تحتوي على ١٠ صفوف ، ٥٠ عموداً فان ذلك يمكن تنفيذه باحدى الطرق – التالية :

طريقة أولى : باعطاء الحاسب كل قيمة في سطر عن طريق الصفوف كالتالي :

```
      DIMENSION X (10,50)
      DO 2 I = 1,10
      DO 2 J = 1,50
2     READ (5,3) X (I,J)
3     FORMAT (20F4.2)
```

في هذه الطريقة نلاحظ أن :

١ - تبعاً لخاصية حلقات التنفيذ المتداخلة ، سنجد أن الدليل J (رقم العمود) سيأخذ القيم من ١ الى ٥٠ عند كل قيمة ثابته من الدليل العددي I (رقم الصف) .

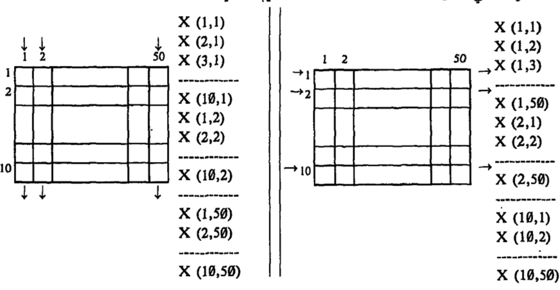
٢ - مع أن الجملة الشارحة تسمح بقراءة ٢٠ قيمة في كل سطر ، الا أن تعليمة القراءة ستجبر الحاسب أن يقرأ قيمة واحدة يتحدد موقعها بقيمتي الدليلين J,I .

طريقة ثانية : باعطاء الحاسب كل قيمة في سطر عن طريق الأعمدة كالتالي :

```
      DIMENSION X (10,50)
      DO 2 J = 1,50
      DO 2 I = 1,10
2     READ (5,3) X (I,J)
3     FORMAT (20 F 4.2)
```

وفي هذه الطريقة قد يتبين الفارق بسهولة عن الطريقة السابقة حيث أن كل ما تم عمله هو تبديل حلقتي التنفيذ بحيث يجب أعطاء الحاسب القيم التي في العمود الأول قيمة بعد أخرى يتلوها قيم العمود الثاني قيمة بعد أخرى أيضاً وهكذا حتى العمود رقم ٥٠ .

ولذا ففي الطريقتين السابقتين يجب إعطاء القيم كالتالي :



شكل (٧-١١)

طريقة ثالثة : باعطاء الحاسب أكثر من قيمة في كل سطر عن طريق الصفوف كالتالي :

```
    DIMENSION X (10,50)
    DO 2 I = 10
2   READ (5,3) (X(I,J), J = 1,50)
3   FORMAT (20 F4.2)
```

وتعتبر هذه الطريقة الأكثر شيوعاً وإنتشاراً في التعامل مع المصفوفات ذوات البعدين ، حيث يتم فيها كتابة قيم كل صف في المصفوفة في سطر واحد أو مجموعة من السطور . ففي مثالنا هذا يجب أعطاء قيم كل صف في المصفوفة على ثلاثة أسطر بحيث يحتوي كل من السطرين الأولين على ٢٠ قيمة والسطر الثالث على ١٠ قيم فقط . أي أنه لكل قيمة جديدة يأخذها الدليل العددي I (والذي يمثل رقم الصف) يجب قراءة ٥٠ قيمة ، حيث أن الدليل العددي J (والذي يمثل رقم العمود) يتغير من ١ الى ٥٠ .

أي أن قيم المصفوفة في تلك الطريقة تعطي كالتالي :

عناصر الصف الأول :

X (1,1)   X (1,2)   ....   X (1,20)
X (1,21)   X (1,22)   ....   X (1,40)
X (1,41)   X (1,42)   .... X (1,50)

عناصر الصف الثاني :

X (2,1)   X (2,2)   ....   X (2,20)
X (2,21)   X (2,22)   ....   X (2,40)
X (2,41)   X (2,42)   .... X (2,50)

عناصر الصف العاشر :

X (10,1)   X (10,2)   ....   X (10,20)
X (10,21)   X (10,22)   ....   X (10,40)
X (10,41)   X (10,42)   .... X (10,50)

شكل (٧-١٢)

ويتضح في هذه الطريقة أن القيمة الأولى من كل صف في المصفوفة تكتب في سطر جديد تتلوها بقية قيم هذا الصف .

طريقة رابعة : باعطاء الحاسب أكثر من قيمة في كل سطر عن طريق الأعمدة كالتالي :

```
    DIMENSION X (10,50)
    DO 2 J = 1,50
2   READ (5,3) (X(I,J) , I=1,10)
3   FORMAT (20 F4.2)
```

ويتبين من تلك الطريقة أنها مشابهة للطريقة الثالثة ، غير أن الحاسب يقوم بقراءة ١٠ قيم تمثل محتويات العمود رقم J الذي يتغير من ١ الى ٥٠ . ولذا فان قيمة المصفوفة في تلك الطريقة تعطي كالتالي :

عناصر العمود الأول :
X (10,1) .... X(3,1) X (2,1) X (1,1)

عناصر العمود الثاني :
X (10,2) .... X(3,2) X (2,2) X (1,2)

عناصر العمود الخمسون :
X (10,50) .... X(3,50)) X(2,50) X (1,50)

ويتضح من هذه الطريقة أن القيمة الأولى من كل عمود تكتب في سطر جديد تتلوها بقية قيم هذا العمود . كما يتضح في مثالنا هذا أننا لن نستطيع إعطاء الحاسب أكثر من ١٠ قيم في كل سطر ، مع أن الجملة الشارحة تسمح لنا باعطائه ٢٠ قيمة ولكن الدليل العددي I في تعليمة القراءة يتغير من ١ الى ١٠ فقط وبعدها يتغير الدليل العددي J والذي يعطي الاشارة للحاسب بأن يبدأ في قراءة قيم عمود جديد وهكذا .

طريقة خامسة : باعطاء الحاسب قيم المصفوفة في مجموعات متساوية في كل سطر (عن طريق الصفوف) .

أي أنه في هذه الطريقة سيتم إعطاء الحاسب قيم الصف الأول في سطر أو أكثر ، ثم نكمل مباشرة قيم الصف الثاني ثم نتبعها بقيم الصف الثالث وهكذا بحيث يحتوي كل سطر على مجموعة قيم معينة (٢٠ قيمة في مثالنا الحالي) لاتزيد ولا تنقص ، لأنه في حالة زيادتها فأن الحاسب لن يشعر بها . كما أنه في حالة نقصها فان الحاسب سيعتبر أن القيم المتبقية تساوي اصفاراً مما قد يسبب أخطاءاً في نتائج الحسابات قد يصعب اكتشافها بسهولة .

وتكتب صيغة القراءة في هذه الطريقة كالتالي :

```
      DIMENSION X (10,50)
      READ (5,3) (X(I,J), J=1, 50), I=1, 10)
3     FORMAT (20 F4.2)
```

ويمكن اعتبار أن تعليمة القراءة في هذه الطريقة تحتوي على حلقتي تنفيذ متداخلتين ، إحداهما خارجية والدليل العددي لها هو المتغير الصحيح I (الذي يمثل رقم الصف في المصفوفة) ، والأخرى داخلية والدليل العددي لها هو المتغير الصحيح J (والذي يمثل رقم العمود في المصفوفة) . ويقوم الحاسب في هذه الطريقة بقراءة قيم المصفوفة جميعها بقسمة كل ٢٠ قيمة على سطر . أي أن قيم المصفوفة في تلك الطريقة يتم اعطاؤها كالتالي :

```
X (1,1)    X (1,2)     .........................................    X (1,20)
X (1,21)   X (1,22)    .........................................    X (1,40)
X (1,41)   X (1,42)    .........X (1,50) X(2,1).........            X (2,10)
X (2,11)   X (2,12)    .........................................    X (2,30)
X (2,31)   X (2,32)    .........................................    X (2,50)
---------------------------------------------------------------------------------
X (10,31)  X (10,32)   .........................................    X (10,50)
```

شكل (٧-١٣)

طريقة سادسة : باعطاء الحاسب قيم المصفوفة في مجموعات متساوية في كل سطر (عن طريق الأعمدة) :

وهي الطريقة المشابهة للطريقة السابقة ولكن بتعديل بسيط في تعليمة القراءة ، وذلك بجعل حلقة التنفيذ التي يمثلها الدليل العددي I كحلقة تنفيذ داخلية وحلقة التنفيذ الأخرى التي يمثلها الدليل العددي J كحلقة تنفيذ خارجية ، مع الأخذ في الاعتبار القيم التي يتغير خلالها كلاً من الدليلين . ولذا فان صيغة القراءة في هذه الطريقة يمكن كتابتها كالتالي :

```
      DIMENSION X (10,50)
      READ (5,3) ((X(I,J), I=1, 10), J=1, 50)
3     FORMAT (20 F 4.2)
```

ولذا فان قيم المصفوفة في تلك الطريقة ستعطي للحاسب بحيث يكتب له قيم كل عمودين في سطر واحد كالتالي :

عناصر العمودين ١، ٢
X (1,1) X (2,1) ..... X (1Ø,1) X (1,2) X (2,2) ... X (1Ø,2)

عناصر العمودين ٣، ٤
X (1,3) X (2,3) ..... X (1Ø,3) X (1,4) X (2,4) ... X (1Ø,4)

X (1,49) X (2,49) ..... X (1Ø,49) X (1,5Ø) X (2,5Ø) ... X (1Ø,5Ø)

وأخيراً كنوع من التدقيق والمقارنة بين الطرق السابقة نجد أنه يجب أن يتم إعطاء جميع قيم المصفوفة X في مثالنا السابق في عدد من الأسطر كالتالي :

الطريقة	عدد الأسطر
الأولى	٥٠٠ (كل قيمة في سطر عن طريق الصفوف)
الثانية	٥٠٠ (كل قيمة في سطر عن طريق الأعمدة)
الثالثة	٣٠ (كل صف في ثلاثة أسطر)
الرابعة	٥٠ (كل عمود في سطر)
الخامسة	٢٥ (كل سطر يحتوي على ٢٠ قيمة مأخوذة عن طريق الصفوف)
السادسة	٢٥ (كل عمودين في سطر يحتوي على ٢٠ قيمة) .

مثال (٣) :

المصفوفة التالية سبق تخزينها في الحاسب ويراد كتابة قيم كل عمود فيها في سطر . اكتب تعليمتي طباعة مختلفتين لتحقيق ذلك .

$$\begin{Bmatrix} 3.07 & -2.1 & 6.75 & -14.8 \\ -6.08 & 7.0 & -0.52 & 3.07 \\ 1.4 & -6.01 & 3.45 & 7.15 \end{Bmatrix}$$

أولاً : لنفترض أن المصفوفة السابقة سبق تخزينها تحت اسم المتغير X .

ثانياً : بالنظر في قيم المصفوفة نجد أن اكبر عدد فيها يحتوي على رقمين صحيحين بالاضافة الى الاشارة ، كما أن أقصى عدد من الأرقام العشرية يحتوي – عليها أي عنصر هو رقمين عشريين فقط . وبالتالي فإن كل عنصر في المصفوفة سيأخذ الصورة F 6.2 .

ثالثاً : لكتابة قيم كل عمود في المصفوفة في سطر ، فإنه يمكن إتباع إحدى الطريقتين التاليتين :

```
C         METHOD A
          DO 7 J=1, 4
     7    WRITE  (6,8) (X(I,J), I=1,3)
     8    FORMAT (12 F 6.2)
```

وفي هذه الطريقة مع أن الجملة الشارحة تسمح بكتابة جميع قيم المصفوفة في سطر واحد ، الا أن حلقتي التنفيذ ذوات الدليلين J,I بهذا الوضع سيجبر الحاسب على تغيير قيم الدليل I عند كل قيمة ثابتة من J ، أي ستمكن الحاسب من كتابة قيم كل عمود في سطر .

```
C         METHOD B
          WRITE (6,9) ((X(I,J), I=1,3), J=1,4)
     9    FORMAT (3 F 4.2)
```

في هذه الطريقة ، فإن تعليمة الطباعة تمكن الحاسب من طباعة جميع قيم المصفوفة عمودا وراء الآخر في سطر واحد ولكن الجملة الشارحة لن تمكن الحاسب من طباعة سوى ثلاث قيم في كل سطر .

مثـال (٤) :

في المصفوفة السابقة اكتب ثلاث تعليمات مختلفة تمكن الحاسب من قراءة قيم المصفوفة قيمة بعد أخرى عن طريق الصفوف .

بعد ما تقدم فانه يمكن كتابة تلك التعليمات كالتالي :

```
C         METHOD 1
          DO 3 I=1, 3
          DO 3 J=1, 4
     3    READ (5,4) X (I,J)
     4    FORMAT (F 4.2)
```

```
C         METHOD 2
          DO 3 I=1, 3
     3    READ (5,4) (X(I,J), J=1,4)
     4    FORMAT (F 4.2)
```

```
C         METHOD 3
          READ (5,4) ((X(I,J), J=1,4), I=1,3)
     4    FORMAT (F 4.2)
```

وفي الطريقة الأولى فسواء كانت الجملة الشارحة في صورتها الحالية أم مكتوبة بصورة أخرى (20 F 4.2 مثلاً) ، فان الحاسب لن يقرأ اكثر من قيمة في كل سطر . اما في الطريقتين الثانية والثالثة فلا بد أن تكون الجملة الشارحة بوضعها الحالي (أي قيمة واحدة في كل سطر) لأن تعليمات القراءة في هاتين الطريقتين تسمحان بقراءة اكثر من قيمة في كل سطر ولن يمنع الحاسب من تنفيذ ذلك سوى الجملة الشارحة في هذه الحالة .

« تمـــارين عـامــة »

١ – ماهو الفرق بين المصفوفة والمتغير العادي من حيث المضمون والشكل العام ؟

٢ – كيف يمكننا اشعار الحاسب بأننا نتعامل مع مصفوفة باسم X طولها 22 ؟

٣ – هل يجوز لنا إستعمال مصفوفة طولها 1 فقط ؟

٤ – ما نوع المقادير الجبرية التي يمكننا إستعمالها كدليل عددي لمصفوفة ؟

٥ – هل يجوز لدليل عددي أن يكون هو نفسه عنصراً في مصفوفة أخرى ، مثل : X(N(J)) ؟

٦ – بين الخطأ في في كل جملة من الجمل التالية :

```
DIMENTION X (5Ø)
A (Ø) = -93
K (K) = NO
A (B/6.7) = 12.Ø
B (1*1.) = 13.Ø
```

٧ – كتب أحد الطلبة البرنامج التالي لقراءة ٥٠ رقماً من ٥٠ سطراً وحساب معدل هذه الأرقام . أوجد اكبر عدد من الأخطاء في هذا البرنامج :

```
      DO 2Ø K = 1,5Ø
         READ (5,1Ø) X (K)
1Ø       FORMAT (F 1Ø.Ø)
2Ø    CONTINUE
      DO 3Ø K = 1,5Ø
         SUM = SUM + X (K)
3Ø    CONTINUE
      AVER = SUM/5Ø.Ø
      WRITE (6,4Ø) AVER
      END
```

٨ – بدون إستعمال الحلقات المباشرة ، اكتب جمل ادخال أو إخراج بيانات مكافئة للجمل التالية على فرضية أن : K = 2, N = 13, M = 4

```
READ (5,501) (X(J) , J = 1,5)
WRITE (6,601) (B(L) , L = M,N,K)
WRITE (6,609) (A(J) , J = K,N,M)
READ (5,20) ((B(I,J) , I = 1,M) , J = 1,K)
WRITE (6,402) Q,R, (S,B(3,B(3,J),A(J),J = 1,K),BC,(A(J),J = 1,4)
```

٩ – إستعمل الحلقات المباشرة لكتابة جمل إدخال وإخراج مكافئة للجمل التالية :

```
WRITE (6,101) A(1), A(2), A(3), A(4), A(5)
READ (5,706) A(3), A(6), A(9), A(12), A(15)
WRITE (6,292) B(2,1), B(3,1), B(2,2), B(3,2), B(2,3), B(3,3)
READ (5,500) A(2), B(3), A(4), B(4), A(6), B(7)
WRITE (6,260) X(1), Y(4), X(2), Y(7), X(3), Y(10)
```

١٠– ماهي الجمل الغير صحيحة بين جمل الادخال والاخراج التالية :

```
WRITE (6,2000) (A(J), J = 1,N - 1)
READ (5,5000) (J,A(J) , J = 1,N)
WRITE (6,6000) (A(J + 1) , J = 1, C (N))
READ (5,501) (A(I), I = - 1,3,2)
WRITE (6,66) A (J)
```

١١– استعمل جملة DIMENSION واحدة لأشعار الحاسب بأن X مصفوفة طولها 25 و Y مصفوفة طولها 36 بينما Z مصفوفة طولها 15 وجميعها ذات قيم حقيقية .

١٢– لنفرض أن A اسم لمصفوفة تتكون من عشر عناصر . اكتب أجزاء من برامج مختلفة لانجاز الآتي :

(أ) ضع ناتج حاصل ضرب العنصر الأول في العنصر الثاني في خلية اسمها PROD .

(ب) إستبدل قيمة العنصر الثالث بمعدل العناصر الأولى والثالثة والخامسة .

(جـ) إذا كان العنصر الأخير مساوياً للصفر أو موجب ، دعه كما هو ، أما حالة كونه سالباً فاعكس إشارته .

(د) بإستعمال حلقة DO إستبدل كل عنصر بضعفه .

(هـ) رتب الثلاثة عناصر الأولى ترتيباً تنازلياً .

١٣- المصفوفة B طولها 20 . اكتب أجزاء من برامج مختلفة لتحقيق التالي :

(أ) ضع حاصل قسمة العنصر الرابع على مجموع العنصرين الخامس والسادس في خلية اسمها ABC .

(ب) إستبدل قيم العناصر الأربع الأخيرة بصفر دون إستعمال حلقة DO .

(جـ) اذا كان العنصر العاشر اكبر من 10 ، إستبدله بمعدل العنصرين التاسع والحادي عشر .

(د) إستبدل قيم العناصر الفردية بالقيمة 1- وإستبدل قيم العناصر الزوجية بالقيمة الثابتة 1 .

(هـ) رتب الأربعة عناصر الأخيرة ترتيباً تصاعدياً .

١٤- ALI و BADR إسمين لمصفوفتين طول كل منها 10 . إستبدل قيمة كل عنصر في ALI بقيمة نظيره من BADR والعكس . ضع مجموع عناصر ALI في خلية اسمها SUM وحاصل ضرب عناصر BADR في خلية اسمها PROD .

١٥- A و B مصفوفتين طول كل منهما 15 اكتب برنامجاً يوجد مجموع حاصل ضرب كل عنصر من A في نظيره من B ، ثم أوجد قيمة الجذر التربيعي للناتج . (اذا كان موجبا !) .

١٦- X و Y مصفوفتين طول كل منها 30 . اكتب برنامجاً يقارن كل عنصر في X مع نظيره من Y ويقوم بطبع رسالة خاصة لكل حالة دون طباعة قيمتى X و Y .

مثـال : X(3) IS GREATER THAN Y(3)

١٧- اكتب برنامجاً يقوم بمهمة ترتيب عناصر مصفوفة اسمها NUMBER وطولها 1000 ترتيباً تصاعدياً .

ملاحظة : هذا البرنامج يصلح لترتيب الطلبة حسب أرقامهم مثلا .

١٨- اكتب برنامجاً ينتج جدول ضرب من حجم ١٢×١٢ .

١٩- اكتب برنامجاً يقوم بمهمة إيجاد اكبر عنصر وأصغر عنصر في مصفوفة طولها 50 .

٢٠- ادى ٩٦ طالباً امتحانا في مادة الرياضيات ١٠١ ع . اكتب برنامجاً ينتج جدولاً بسيطاً يوضح توزيع الدرجات كما هو موضح في المثال التالي :

GRADE RANGE	NUMBER OF STUDENTS
90-100	15
80-89	21
70-89	29
60-69	16
0-59	15

٢١- من المقادير الشائعة في العمليات الاحصائية ، المعدل Mean والانحراف القياسي Standand Deviation أما المعدل فيمكن ايجاده باستعمال القانون :

$$\bar{X} = \left(\sum_{i=1}^{n} X_i \right) / n$$

وأما الانحراف القياسي فيمكن ايجاده بالقانون :

$$S = \left(\sum_{i=1}^{n} (X_i - \bar{X})^2 / n-1 \right)$$

اكتب برنامجاً لايجاد هذين المقدارين لعينة احصائية تتكون من ٢٠ قراءة .

٢٢- يقوم أحد طلاب مادة الأحياء بدراسة لبعض فئات من الحشرات الصغيرة الموزعة الى عشرين صنفاً تحمل أرقاماً من ١-١٠ . أحد خطوات التجربة تحتاج الى تسجيل وزن الحشرة وانتاج جدول يظهر معدل وزن حشرات كل صنف .
البيان المعطى مكون من عدة مئات من السطور ، يتكون كل سطر منها من قراءتين القراءة الأولى لرقم الصنف في الخانتين الأولى والثانية . أما القراءة العشرية الثانية في الخانات من ٣ الى ٨ وتمثل وزن الحشرة بالجرام (بما في ذلك الفاصلة العشرية) .
اكتب برنامجاً يقرأ هذه السطور حتى يجد السطر المزيف حيث رقم الصنف يكون خارج نطاق (١ الى ٢٠) وهذا بمثابة إشعار بانتهاء البيان المعطى . المطلوب ايجاد عدد الحشرات من كل صنف ومجموع أوزان كل صنف ومن ثم معدل وزن كل صنف .

٢٣- جمع طالب في مادة علم النفس مئات من الدرجات التي استحقها طلاب مختلفون من الجامعة في ثلاثين مادة مختلفة تحمل كل مادة رقم كمبيوتر خاص بها . المطلوب كتابة تقرير يوضح العلامة الكبرى والعلامة الصغرى في كل مادة على حده .

اكتب برنامجاً ينتــج بياناً مكوناً من ثلاثين سطراً يحتوي كل سطر على رقم المـادة وعدد الطلاب الذين تقدموا للامتحان فيها بالاضافة الى العلامة الصغرى والكبرى .

٢٤- اكتب برنامجاً يقوم بمهمة الحجز الآلي لخطوط طيران الجامعة التي يتكون أسطولها من سبع طائرات تقوم يومياً بسبع رحلات بسعات مختلفة كما يلي :

رقم الرحلة	السعة راكب
٠٠١	١٥٠
٧٠٧	١٣٢
١٠٦	٢٠٨
٢٠٩	١٣٠
٠٦٢	١٦٦
٢٥٠	١٥٩
٣٠٣	١٧٠

علماً بأن الحجز يتم بطريقة الأول فالأول « بدون واسطة » . المطلوب انتاج بيانين مختلفين يحتوي الأول على رقم الرحلة ثم أسماء المسافرين عليها وأرقام هواتفهم أما البيان الثاني فيحول اسم جميع الأشخاص الحاجزين وأرقام هواتفهم والرحلة التي تم الحجز عليها .

الفصل الثامن

البرامج الجزئية

الفصـل الشـامن

الـبرامـج الجـزئيـة Subprograms

مـقـدمـة :

تمتاز لغة الفورتران عن معظم لغات الحاسبات بامكانية استخدام مايسمى « بالبرامج الجزئية » حتى أن بعض لغات الحاسبات الأخرى بدأت في تطوير لغاتها باضافة هذا النوع من البرامج الجزئية الى لغاتها الأصلية . وفي الفصل السادس رفعنا شعار « كل برنامج يمكن تصغيره أو اختصاره » ، ولعل استخدام البرامج الجزئية يعتبر أقوى سلاح لتحقيق هذا الهدف . ففي أغلب المشاكل العلمية قد يضطر المبرمج الى تكرار كتابة مجموعة من التعليمات أكثر من مرة في نفس البرنامج ، مما يجعل البرنامج طويلاً ومملاً ويكون عرضة لكثير من الأخطاء . وفي عصرنا الحالي عرفنا طرقاً لحلول مشكلات مختلفة وأطلق على تلك الطرق اسم الخوارزم وعند عمل برنامج لحل إحدى تلك المشكلات فان المبرمج سيضطر لكتابة مجموعة من التعليمات لتنفيذ الخوارزم المستخدم في حل مشكلته وقد يضطر لتكرار كتابتها اكثر من مرة في نفس البرنامج ان تطلب الأمر ذلك .

فعلى سبيل المثال ، لحساب التوافيق$(\frac{N}{R})$ بين عددين R,N مثلاً فان الأمر يتطلب حساب مضروب N (N!) ‎, R!, (N−R)!‎ حيث أن :

$$(\frac{N}{R}) = \frac{N!}{R!\ (N-R)!}$$

وذلك قد يتطلب تكرار كتابة مجموعة التعليمات الخاصة بحساب المضروب ثلاثة مرات في نفس البرنامج . واذا تطلب الأمر حساب التوافيق اكثر من مرة في نفس البرنامج فقد يستدعي ذلك تكرار كتابة تلك المجموعة من التعليمات عدداً كبيراً من المرات مما يجعل البرنامج طويلاً ومملاً ومكرراً في اكثر اجزائه وعرضه لكثير من الأخطاء . ولكن إذا أمكن وضع مجموعة التعليمات التي تتكرر اكثر من مرة في برنامج جزئي مستقل يتم تخزينه جانباً في وحدة التخزين ويتم استدعاؤه عن طريق البرنامج الرئيسي كلما تطلب الأمر ذلك فان ذلك لاشك سيوفر الكثير من وحدات التخزين كما سيختصر كثيراً من حجم البرنامج ويجعل التحكم في اجزائه ممكناً وسهلاً . وما ينطبق على مجموعة من التعليمات تتكرر اكثر من مرة في البرنامج ، فان نفس الشيىء قد يحدث لتعليمة واحدة فقط يتم حسابها اكثر من مرة . وفي جميع تلك الحالات فانه من المستحسن إستخدام ما يسمى بالبرامج الجزئية .

وفيما يلي سنستعرض الأنواع المختلفة من البرامج الجزئية وكيفية إستخدامها .

٨-١ الدول سابقة التخزين : Library Functions or Preprogrammed Packages

هناك بعض الدوال التي يكثر استخدامها في حل المشاكل الرياضية والهندسية المختلفة مثل ايجاد الجذر التربيعي لقيمة ما أو ايجاد لوغاريتم عدد ما أو ايجاد جيب زاوية أو جيب تمامها ... الخ . مثل تلك الدوال الهامة والتي قد يتطلب الأمر استخدامها اكثر من مرة في البرنامج أو يتطلب استخدامها في أغلب البرامج المختلفة ، يتم تخزينها مسبقاً في الحاسب عن طريق ماسبق أن اطلقنا عليه اسم المجمعات أو المشغلات Compilers . وتتوقف نوعية تلك الدوال وعددها وطريقة استخدامها على نوع الحاسب وطرازه ، ولذا فيجب على كل مبرمج أن يكون على علم بما يحتويه الحاسب الذي سيعمل عليه من دوال مخزنة . وعموماً فان معظم مشغلات الحاسبات العددية تحتوي على الدوال الآتية :

١ - الجذر التربيعي : Square root

عند حساب الجذر التربيعي لأي عدد أو تعبير رياضي فان ذلك يتم ببساطة وبدون أن يبذل المبرمج أي مجهود في كتابة مجموعة التعليمات(*) التي يتطلبها حساب الجذر التربيعي وذلك عن طريق استدعاء البرنامج الجزئي المكلف بتلك المهمة والذي يكون قد سبق تخزينه في موضع ما من وحدة التخزين للحاسب وذلك باستخدام التعبير **SQRT (a)** حيث a تمثل ثابت حقيقي أو متغير حقيقي أو تعبير رياضي يحوي مجموعة من المتغيرات الحقيقية .

وبشرط أن a يجب أن تكون موجبة .

مثال (١) : **Y = SQRT (X)**

في هذا المثال ، فان الحاسب سيقوم بحساب الجذر التربيعي للمتغير X ثم يقوم بتخزين تلك القيمة في المتغير Y وذلك بإفتراض أن قيمة X غير سالبة .

مثال (٢) : **ROOT = SQRT (B*B – 4.0*A*C)**

بعد حساب التعبير الرياضي (B*B – 4.0*A*C) وايجاد قيمته ، يقوم الحاسب باختبار ما اذا كانت تلك القيمة موجبة أم سالبة ، وفي حالة ما اذا كانت موجبة يتم حساب الجذر التربيعي لتلك القيمة وتخزن في المتغير ROOT ، اما اذا كانت سالبة فغالباً ما يعطي الحاسب اشارة خطأ (Error) ويتوقف عن تكملة الحسابات .

(*) من المعروف في الحسابات العددية أن هناك طرقاً رياضية كثيرة لحساب أي دالة ، وتمتاز كل طريقة عن الأخرى في درجة دقتها وسرعة الحساب بها .

٢ – القيمة المطلقة : **Absolute Value**

تقوم هذه الدالة عند استخدامها في البرنامج بعمل ما يدل عليه اسمها تماماً ، أي أنها تقوم بإهمال إشارة قيمة المتغير المطلوب ايجاد قيمته المطلقة . والصورة العامة لهذه الدالة : **IABS (K) , ABS (X)** حيث X تمثل متغير حقيقي أو تعبير رياضي له قيمة حقيقية ، بينما K تمثل متغير صحيح أو تعبير رياضي له قيمة صحيحة .

مثال : **ROOT = SQRT (ABS (B*B – 4.Ø*A*C))**

ففي هذا المثال ، فان القيمة التي ستخزن في المتغير ROOT يتم حسابها بعد أن يقوم الحاسب بمعرفة القيمة المطلقة للتعبير (B*B – 4.Ø*A*C) وأخذ الجذر التربيعي لتلك القيمة . ويلاحظ في المثال السابق أن لكل دالة مجموعة الأقواس الخاصة بها ، ويقوم الحاسب بفك تلك الأقواس طبقاً للطريقة التي سبق شرحها في الفصل الثاني .

٣ – الدالة الأسية(*) **Exponential**

كثيراً ما نستخدم في حساباتنا العلمية الدالة الأسية e(2.71828..) والتي عادة ما تكون في الصورة e^x حيث x تمثل ثابت حقيقي أو متغير حقيقي أو كمية حقيقية ، ولذا فان الصورة العامة لتلك الدالة في لغة الفورتران هي **EXP (x)** .

مثال : **Y = A*EXP (2.Ø*A – B**C)**

في هذا المثال يتم حساب المقدار (2.Ø*A – B**C) أولاً ، ثم يتم حساب الدالة الأسية لهذا المقدار ويضرب في قيمة المتغير A ، ثم توضع النتيجة النهائية في المتغير Y .

٤ – دالة اللوغاريتم(*) **Logarithms**

احدى الدوال الأخرى التي يكثر استخدامها في حلول المشاكل الرياضية ، وعن طريق بعض الحاسبات يمكن حساب نوعين مختلفين من تلك الدالة :

١ – دالة اللوغاريتم الطبيعي (أي للأساس e) وصورتها العامة في لغة الفورتران هي

(*) يمكن حساب الدالة الأسية e^x عن طريق حساب مفكوك المتسلسلة :

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\ldots + \frac{x^n}{n!} \quad \ldots\ldots$$

(*) احدى الصيغ الرياضية لدالة اللوغاريتم هي :

$$\text{Log}\,(1-x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots\ldots$$

ALOG (x)

٢ – دالة اللوغاريم العام (أي للأساس ١٠) وصورتها العامة هي **ALOG 10 (x)** وفي كلتا الدالتين فان x تمثل ثابت حقيقي أو متغير حقيقي أو تعبير رياضي يأخذ قيمة حقيقية اكبر من الصفر .

مثـال :

```
R = ALOG (A**2+B**2)
S = ALOG 10 (z)
```

٥ – دوال الجيب وجيب التمام والظل : Sine, Cosine and Tangent

بعض الحاسبات تحوي تلك الدوال الثلاث ، بينما معظمها لايحوي غير دالتي الجيب وجيب التمام . وعموماً فان الصورة العامة لتلك الدوال هي : **SIN (x), COS (x), TAN (x)**

وفي جميع تلك الدوال فان الزاوية x يجب أن يعبر عنها **بالتقدير الدائري** ، واذا لم تكن كذلك فيجب تحويلها أولاً قبل استخدام أي من تلك الدوال . واذا لم يكن الحاسب المستخدم يحوي دالة الظل فأنه يمكن حسابها بسهولة باستخدام القانون : tan (x) = sin (x) / cos (x) ، وبشرط أن x لا تساوي :

$$(n\pi + \frac{\pi}{2})$$

حيث n عدد صحيح موجب اكبر من أو يساوي الصفر حيث أن COS (x) عند تلك الحالات سيساوي صفرا .

مثـال :

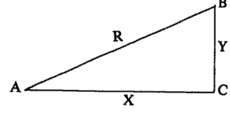
```
Y = SIN (ANGLE)
YD = SIN (XRAD)* COS (YRAD)
YAL = Y* TAN (XRAD)
```

وهناك دوال أخرى بعضها يعتمد على متغير واحد وبعضها الآخر على اكثر من متغير ، وفي الجدول التالي سنذكر بعض أهم الدوال الأخرى التي تكثر استخدامها والتي يحتوي عليها الحاسب I.B.M. / 370

استخدام الدالة	المعنى الرياضي	اسم الدالة	عدد المتغيرات	نوع المتغير	نوع النتيجة
تحويل	التحويل الى قيمة صحيحة	INT	١	حقيقي	صحيح
		IFIX	١	حقيقي	صحيح
	التحويل الى قيمة حقيقية	REAL	١	صحيح	حقيقي
		FLOAT	١	صحيح	حقيقي
حذف المتبقي	الجزاء الصحيح من قيمة متغير	AINT	١	حقيقي	حقيقي
المتبقي	x - int (x/y) *y	AMOD	٢	حقيقي	حقيقي
اكبر قيمة	max (x1, x2 ....)	AMAX 0	اكبر من	صحيح	حقيقي
		AMAX 1	أو يساوي	حقيقي	حقيقي
		MAX 1	٢	حقيقي	صحيح
أقل قيمة	min (x1, x2 ....)	AMIN 0	اكبرمن	صحيح	حقيقي
		AMIN 1	أو يساوي	حقيقي	حقيقي
		MIN 1	٢	حقيقي	صحيح
حساب قيمة زاوية	$\sin^{-1}(x)$	ARSIN	١	حقيقي	حقيقي
	$\cos^{-1}(x)$	ARCOS	١	حقيقي	حقيقي
	$\tan^{-1}(x)$	ATAN	١	حقيقي	حقيقي
	$\tan^{-1}(x/y)$	ATAN 2	٢	حقيقي	حقيقي
دوال زائدية	sinh (x)	SINH	١	حقيقي	حقيقي
	cosh (x)	COSH	١	حقيقي	حقيقي
	tanh (x)	TANH	١	حقيقي	حقيقي

أمثلة على إستخدام بعض الدوال السابقة :

```
SINA = SIN (A)
   B = ARCOS (0.5)
   A = ARSIN (SINA)
   B = ATAN2 (X/Y)
XMIN = AMIN1 (X1, X2, X3, X4)
   R = AMOD (48.5, 6.3)
```



في التعليمة الأخيرة فان قيمة R تمثل الفارق بين 48.5 واكبر قيمة صحية للكسر $\frac{48.5}{6.3}$ مضروبة في 6.3 ، أي أن :

$$R = 48.5 - \text{int}\left(\frac{48.5}{6.3}\right) \times 6.3 = 48.5 - 7 \times 6.3 = 4.4$$

٨-٢ الدوال ذات التعبير الرياضي : Arithmetic Statement Functions

عندما يتكرر حساب تعبير رياضي في برنامج ما ، فان هذا التعبير يمكن أن يوضع في صورة دالة يعطي لها اسم من واضع البرنامج ، وهذا الأسم يختلف عن أسماء الدوال سابقة التخزين التي ورد ذكرها في ٨-١ ، ويتم حساب قيمة تلك الدالة في البرنامج كلما ورد اسمها وقيم المتغيرات التي تعتمد عليها . كما أنه لايمكن اعتبار مثل تلك الدوال برامج جزئية لسببين :

١ - أن التعبير الرياضي الذي يمثل تلك الدالة ، يمكن اعتباره احدى التعليمات الواردة في البرنامج ، مثلها مثل أي تعليمة اخرى ، أي أنها تعتبر جزءاً من البرنامج نفسه .

٢ - أن تلك الدالة تكتب في صورة تعليمة واحدة فقط وليس أكثر ، وبالتالي لايلزم لها أية تعليمة أخرى مثل تعليمة النهاية END Statement التي تشعر الحاسب بأن البرنامج قد أنتهى عند تلك التعليمة .

ويجب أن توضع التعليمة التي تمثل تلك الدالة في أول البرنامج بحيث تسبق أي تعليمة تنفيذية أخرى فيه مثل تعليمات القراءة أو الكتابة أو الحساب .... الخ . والصورة العامة لمثل هذا النوع من الدوال هي :

$$\text{NAME}(a_1, a_2, \ldots a_n) = \text{arithmetic expression in } a_1, a_2 \ldots a_n$$

حيث NAME : يدل على الاسم المعرّف للدالة والذي ينطبق عليه نفس شروط اعطاء اسم لأي متغير ، وقيمة الدالة من حيث أنها حقيقية real أو صحيحة integer يتوقف على اسم الدالة وأول حرف فيه أو من تعريف الدالة نفسها مسبقا .

$a_1, a_2 \ldots\ldots a_n$: متغيرات سيتم استخدامها في حساب قيمة الدالة ويجب أن تكون موجودة داخل التعبير الرياضي المستخدم في حساب قيمة الدالة . وعند حساب الدالة في اكثر من موضع في البرنامج فانه ليس من الضروري أن تستخدم نفس تلك المتغيرات ولكن يمكن استخدام أسماء ومتغيرات أخرى ولكن بشرط أن تكون مساوية لعدد المتغيرات الأصلية ، وكل متغير يناظر في نوعة من حيث أنه حقيقي أو صحيح للمتغير الأصلي المستخدم في حساب قيمة تلك الدالة .

مثـال :

```
      ROOT1 (A,B,C)= (-B+SQRT(B**2-4.Ø*A*C))/(2.Ø*A)
      ROOT2 (A,B,C)=(-B-SQRT (B**2-4.Ø*A*C))/(2.Ø*A)
      READ (3,1Ø) X,Y,Z
1Ø    FORMAT (3F 7.2)
      :

      :
      S = (X + Y + Z) / 3.Ø
      R1 = ROOT1 (S,Y,Z)
      R2 = ROOT2 (S,Y,Z)
      WRITE (6,15) R1, R2
15    FORMAT ('ɃR1 =', F12.3, 5x, 'R2 =', F12.3)
      STOP
      END
```

نلاحظ في المثال السابق :

١ – أننا قمنا بتعريف دالتين مختلفتين أحداهما تسمى ROOT 1 والأخرى تسمى ROOT 2 ولو أن كلاً منهما يعتمد على المتغيرات C,B,A ، وقد تم تعريف هاتين الدالتين وذلك بوضعهما كأول جملتين في البرنامـج ، وقبل أي جملة تنفيذية .

٢ – أنه قد تم إستخدام الدالتين في بعض أجزاء البرنامـج ولكن بمتغيرات أخرى Z,Y,S سبق معرفة قيمها عن طريق تعليمة القراءة مثلاً ، أو عن طريق حسابها في البرنامـج.

ولذا فان قيمة S ستستخدم في الدالتين لتحل محل المتغير A وقيمة Y محل المتغير B وقيمة Z للمتغير C .

٣ – أن التعبير الرياضي للدالة قد تحتوي على دوال سابقة التخزين مثل SQRT في مثالنا هذا .

٤ – حيث أن اسم كل دالة يمثل متغير حقيقي ، لذا فيجب أن نتوقع أن قيمة كل دالة منهما حقيقية أيضا .

٨–٣ الدوال في صورة برامج جزئيـة : Function Subprograms

فيما سبق ذكرنا نوعين من الدوال ، احداهما الدوال سابقة التخزين والتي يكون قد سبق تحضيرها وتخزينها بواسطة الشركة المنتجة للحاسب والتي لا تتطلب جهداً في استخدامها سوى وضع أسم تلك الدالة بمتغيراتها في صورة تعليمة تنفيذية داخـل البرنامـج ، والأخرى دوال ذات تعبير رياضي

توضع في أول البرنامج وكل دالة تأخذ صورة تعليمة تنفيذية طرفها الأيسر يشمل اسم الدالة والمتغيرات التي تعتمد على حسابها تلك الدالة ، بينما طرفها الأيمن يشمل الصيغة الرياضية المطلوب حسابها . ويتم حساب تلك الدالة في أي موضع من البرنامج بوضع اسم الدالة ومتغيراتها ، أو متغيرات نظيرة لها في صورة تعليمة تنفيذية ، بعد اعطاء قيم للمتغيرات الداخلة في حساب تلك الدالة .

وهناك نوع ثالث من الدوال يحتاج لحسابها كتابة اكثر من تعليمة تنفيذية ، وطالما أن الهدف من استخدام تلك الدوال هو الاضطرار الى حسابها اكثر من مرة داخل البرنامج ، لذا فإن هذا النوع من الدوال يكتب في صورة برنامج جزئي مستقل عن البرنامج الأصلي ، وعندما يراد حساب تلك الدالة في البرنامج الأصلي يتم استدعاؤها عن طريق تعليمة تنفيذية تشمل في طرفها الأيمن اسم الدالة والمتغيرات اللازمة لحساب قيمتها .

وعندما ينتقل الحاسب الى البرنامج الجزئي لحساب قيمة الدالة فانه يعود الى البرنامج الأصلي ، الى التعليمة التي تم استدعاؤه منها عن طريق التعليمة RETURN Statement .

والصورة العامة للبرامج الجزئية التي تستخدم لحساب دالة هي :

```
FUNCTION name (a1, a2....., an)
:        :
name = expression
RETURN
END
```

حيث : FUNCTION : كلمة تعريف للحاسب بأن التعليمات التالية تمثل برنامج جزئي لحساب دالة ما ،

name : اسم للدالة المطلوب حسابها ، وقيمة الدالة من حيث أنها قيمة حقيقية أو صحيحة يتوقف على اسم الدالة ، $(a_1, a_2....., a_n)$: متغيرات تلزم لحساب قيمة الدالة ، فاذا كان احداها يمثل مصفوفة فانه فانه يجب وضع تلك المصفوفة في جملة توضيحية DIMENSION أو REAL أو INTEGER قبل أي جملة تنفيذية في البرنامج ،

RETURN : تعليمة للعودة الى البرنامج الأصلي الى الموضع الذي تم استدعاء الدالة منه ،

END : تعليمة نهاية البرنامج الجزئي .

مثـال (١) :

كما ذكرنا في مقدمة هذا الفصل ، فانه لحساب التوافيق C_R^N بين عددين صحيحين R, N فان ذلك يتطلب حساب المضروب ثلاث مرات هي N!, R!, (N − R)! حيث أن :

$$C_R^N = \frac{N!}{R!\,(N-R)!} .$$

ولذا فمن المفيد في هذه الحالة أن نكتب برنامجاً جزئياً لحساب مضروب أي عدد صحيح . وفي البرنامج الأصلي نستدعي ذلك البرنامج الجزئي كلما تطلب الأمر حساب مضروب عدد ما .

```
C       A FUNCTION SUBPROGRAM TO COMPUTE THE
C       FACTORIAL OF
C       A GIVEN + VE INTEGER NUMBER
        FUNCTION IFAC (L)
        IFAC = 1
        IF (L. EQ.Ø) RETURN
        DO 5 I = 2, L
    5   IFAC = IFAC*I
        RETURN
        END

C       MAIN PROGRAM
C       N MUST BE GREATER OR EQUAL K
        READ (5,1) N,K
    1   FORMAT (2I3)
        NN = IFAC (N)
        KK = IFAC (K)
        L = N-K
        LL = IFAC (L)
        KOMB = NN / (KK*LL)
        WRITE (6,2) N,K, KOMB
    2   FORMAT ('ƀN=',I3, 2X, 'ƀK=',
       1I3, 5X,'ƀCOMB =', I6)
        STOP
        END
```

نلاحظ في المثال السابق ما يلي :

١ – اعطينا البرنامج الجزئي الذي سيقوم بحساب دالة المضروب اسم IFAC وذلك كي نضمن أن النتيجة ستكون قيمة صحيحة لاتحتوي على أية كسور .

٢ – في البرنامج الأصلي ، يتم استدعاء البرنامج الجزئي عن طريق تعليمة تنفيذية يذكر في طرفها الأيمن اسم الدالة والمتغيرات الداخلة فيها ، وليس من المهم أن تكون اسماء تلك المتغيرات هي نفس اسماء متغيرات دالة البرنامج الجزئي ولكن المهم أن تكون من نفس النوع (صحيحة أو حقيقية) .

٣ – أن البرنامج الجزئي الذي سيقوم بحساب الدالة قد يحوي اكثر من تعليمة RETURN ، ولكنه يحوي تعليمة انهاء البرنامج END مرة واحدة .

٤ – أن قيمة الدالة يتم حسابها في البرنامج الجزئي عن طريق تعليمة حسابية يكون الطرف الأيسر فيها هو اسم الدالة .

مثـال (٢) :

لنفترض أن لدينا ٢٠ مجموعة من الأعداد وكل مجموعة تتكون من ٥ أعداد قد تحتوي على أعداد سالبة . ويراد حساب الوسط الحسابي لكل مجموعة تحوي ٥ أعداد جميعها موجبة . وسنفترض أن البرنامج الرئيسي سيقوم بقراءة مجموعات الأعداد وكتابة الوسط الحسابي للمجموعة التي تفي بالشروط الموضوعة . كما سنفترض أن هناك برنامجين جزئيين لدالتين احداهما تسمى TEST وذلك لأختبار أعداد كل مجموعة وهل جميعها اكبر من الصفر أم لا ، فإن كانت المجموعة تحوي عدداً سالباً على الأقل فان قيمة تلك الدالة ستساوي صفرا ، بينما الدالة الأخرى وتسمى AVR وتختص بحساب الوسط الحسابي للمجموعة التي تفي بالشروط .

```
C       MAIN PROGRAM
        DIMENSION GROUP (5)
        REAL MEAN
        DO 10 I=1, 20
        READ (2,5) (GROUP (J) , J=1, 5)
5       FORMAT (5F6.2)
        MEAN = TEST (GROUP)
10      WRITE (6,15) (GROUP (J) , J=1, 5), MEAN
15      FORMAT (5F7.2, 5X, 'AVERAGE =', F7.2)
        STOP
        END
```

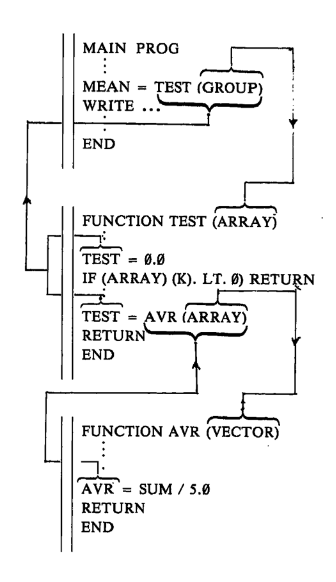
3.	2.	−4.	5.	7.	1
6.	3.	5.	1.	4.	2
⋮		⋮			
					20

```
C        A FUNCTION SUBPROGRAM TO TEST A GROUP OF NOS.
         FUNCTION TEST (ARRAY)
         DIMENSION ARRAY (5)
         TEST = 0.0
         DO 10 K = 1,5
         IF (ARRAY (K). LT. 0.0) RETURN
   10    CONTINUE
         TEST = AVR (ARRAY)
         RETURN
         END
```

```
C        FUNCTION SUBPROGRAM TO COMPUTE AVERAGE
         FUNCTION AVR (VECTOR)
         DIMENSION VECTOR (5)
         SUM = 0.0
         DO 5 I = 1,5
    5    SUM = SUM + VECTOR (I)
         AVR = SUM / 5.0
         RETURN
         END
```

في هذا المثال نلاحظ أن :

١ - البرنامج الأصلي يستدعي برنامجاً جزئياً لحساب الدالة TEST ، وأن البرنامج الذي يقوم بحساب تلك الدالة يقوم بإستدعاء برنامجاً جزئياً آخر لحساب الدالة AVR ، ولذا فان تسلسل العمليات سيكون كالتالي (أنظر الشكل) .



(أ) قراءة مجموعة الأعداد عن طريق البرنامج الأصلي والمخزونة في المصفوفة GROUP .

(ب) الانتقال بتلك الأعداد الى البرنامج الجزئي للدالة TEST ،

(جـ) في البرنامج الجزئي للدالة TEST يتم التعامل مع مجموعة الأعداد في مصفوفة تحت اسم ARRAY ،ويُجرى اختبار على مجموعة الأعداد تلك فان كانت تحتوي أية أعداد سالبة فان الدالة تأخذ القيمة صفرا ثم تنتقل الحسابات الى البرنامج الأصلي لكي يتم طباعة مجموعة الأعداد تلك وقيمة الدالة TEST ، أما اذا كانت جميع الأعداد موجبة فان الحسابات تنتقل الى البرنامج الجزئي للدالة AVR حيث يعبر اسم الدالة عن قيمة الوسط الحسابي لتلك المجموعة من الأعداد التي يتم التعامل معها في مصفوفة أخرى تحت اسم VECTOR .

(د) بعد أن يتم حساب قيمة الدالة AVR ، تنتقل الحسابات مرة أخرى إلى البرنامج الجزئي للدالة TEST كي تكون قيمة تلك الدالة معبرة عن قيمة الوسط الحسابي لمجموعة الأعداد المعطاه .

(هـ) في نهاية البرنامج الجزئي للدالة TEST تعود الحسابات مرة أخرى الى البرنامج الرئيسي والذي يتم فيه وضع قيمة الدالة TEST (الوسط الحسابي) في المتغير MEAN .

(و) في نهاية البرنامج الرئيسي يتم كتابة مجموعة الأعداد وقيمة وسطها الحسابي ، ثم يتوقف البرنامج الرئيسي عن الحساب .

٢ – البرنامجين الجزئيين للدالتين تحويان في كل منها التعليمة الوصفية DIMENSION ، وأن عدد الخلايا المخصصة لكل مصفوفة في الدالتين يساوي العدد المخصص للمصفوفة GPOUP وأن اسم المصفوفة بدون وصف أو بُعد هو الذي يظهر كأحد متغيرات الدالة .

٣ – ومع أن البرنامج الأصلي مرتبط مع البرنامجين الجزئيين للدالتين ، الا أن كل برنامج من البرامج الثلاث يمكن اعتباره برنامجاً مستقلاً بأسماء المتغيرات وأرقام التعليمات التي فيه وكلاً منها يحتوي على تعليمة النهاية END .

٤ – ان النتيجة النهائية لكل برنامج جزئي لحساب دالة عبارة عن **قيمة واحدة فقط** وليس اكثر تمثل قيمة تلك الدالة ، وقد يكون هذا هو أحد الاختلافات الجوهرية بين البرامج الجزئية للدوال والبرامج الجزئية الفرعية SUBROUTINE Subprograms التي سنتحدث عنها الآن .

٨–٤ البرامج الجزئية الفرعية SUBROUTINE Subprograms

احدى الوسائل الأخرى الفعالة في اختصار برنامج يحتوي على مجموعة من التعليمات الحسابية المتكررة هي استخدام ما يعرف بالبرامج الجزئية الفرعية . وهناك بعض أوجه الشبه بين هذا النوع من البرامج وبين البرامج الجزئية للدوال ، منها :

١ – أن كلاً منهما يعتبر برنامجاً مستقلاً عن البرنامج الأصلي ، في أسماء متغيراته وأرقام التعليمات التي يحتوي عليها ولا يربط بينهما وبين البرنامج الأصلي سوى ورود اسم البرنامج الجزئي في صورة تعليمة في البرنامج الأصلي .

٢ – أن كلاً من النوعين يحمل اسماً يميزه عن بقية البرامج الجزئية الأخرى ، كما أنه عن طريقه يتم استدعاؤه الى البرنامج الأصلي . والصورة العامة للتعليمة الأولى في البرامج الجزئية الفرعية هي :

```
SUBROUTINE name
SUBROUTINE name (a1, a2....., an)        أو
```

والشروط الموضوعة على اعطاء الاسم للبرنامج الجزئي الفرعي هي نفس الشروط بالنسبة لأي متغير ، أي أن الاسم لايزيد عن ستة حروف وأرقام بحيث يكون أوله حرفا . **والاسم هنا لايدل على نوع النتائج** التي سيُكلف البرنامج الجزئي الفرعي بحسابها ، فقد يكون الاسم صحيحاً Integer بينما تكون النتيجة في صورة أعداد حقيقية . وقد تكون هذه إحدى فقط الأختلاف بين البرامج الجزئية للدوال والبرامج الجزئية الفرعية .

٣ - أن كلاً منهما يحتوي بداخله على تعليمة عودة RETURN Statement أو اكثر الى البرنامج الأصلي ، كما أن كلاً منهما يحتوي على تعليمة نهاية END Statement واحدة لانهاء البرنامج الجزئي . وبالتالي فان الصورة العامة للبرامج الجزئية الفرعية ككل هي :

```
SUBROUTINE name (a1, a2....., an)
    :
    :        }      مجموعة تعليمات بلغة الفورتران
RETURN
END
```

حيث : name : اسم البرنامج الجزئي الفرعي ، ولا يشترط هنا أن يكون الاسم دالاً على نوع النتائج (حقيقية أم صحيحة) التي سيقوم بحسابها ، كما لا يجب استخدام اسم البرنامج الجزئي الفرعي كمتغير في البرنامج الأصلي .

$(a_1, a_2 \ldots\ldots, a_n)$: متغيرات يتم استخدامها في البرنامج الجزئي الفرعي **وبعض** تلك المتغيرات يكون قد سبق اعطاؤها قيماً في البرنامج الأصلي **وبعضها** الآخر قد يستخدم كنتائج لحسابات البرنامج الجزئي الفرعي ، كما أن بعض تلك المتغيرات قد تمثل مصفوفات ذات بعد واحد أو بعدين . فمثلاً لنفرض أن البرنامج الجزئي الفرعي يقوم بحساب مجموع مصفوفتين Y,X كلاً منهما ذات بعدين $(M \times N)$ ، فان التعليمة الأولى في البرنامج الفرعي ستكون :

```
SUBROUTINE MATADD (M,N,X,Y,Z)
DIMENSION X (M,N), Y (M,N), Z (M,N)
```

وذلك بافتراض أن المصفوفة Z هي التي سيتم فيها جمع المصفوفتين Y, X .

وكما في حالة البرامج الجزئية للدوال ، يجب تعريف أبعاد ثلاث مصفوفات مختلفة في البرنامج الأصلي ، وكل منها ذات بعدين يساوي N,M ، ولا يهم ان كانت اسماؤها هي نفس أسماء مصفوفات البرنامج الجزئي الفرعي أم لا . كذلك قد لايحتاج البرنامج الجزئي الفرعي الى أية متغيرات . وعلى سبيل المثال ، فان البرنامج الجزئي الفرعي التالي لايقوم مثلاً الا بكتابة عنوان ما كلما يتم استدعاؤه عن طريق البرنامج الأصلي ، ولذا لايتطلب أية متغيرات لعمل أية حسابات .

```
      SUBROUTINE TITLE
      WRITE (6,5)
5     FORMAT ('1', 30X, 'ADDITION OF TWO MATRICES' / 30X, 24 (1H=) //)
      RETURN
      END
```

: تمثل أية تعليمات بلغة الفورتران سواءً كانت تلك التعليمات حسابية أو منطقية أو تعليمات انتقال .. الخ . ولكن يشترط الا تكون بين تلك التعليمات تعليمتى FUNCTION, SUBROUTINE . وقد يمكن من خلال تعليمات البرنامج الجزئي الفرعي استدعاء برنامج جزئي لحساب دالة أو برنامج جزئي فرعي آخر .

RETURN : تعليمة العودة الى البرنامج(*) الذي تم منه استدعاء البرنامج الجزئي الفرعي . وقد يحتوي البرنامج الجزئي الفرعي على اكثر من تعليمة من هذا النوع .

END : تعليمة انهاء البرنامج الجزئي الفرعي .

٨-٤-١ تعليمة استدعاء البرنامج الجزئي الفرعي : CALL STATEMENT

في الأنواع السابقة من الدوال (الدوال سابقة التخزين أو الدوال في صورة تعبير رياضي أو البرامج الجزئية للدوال) ، كان يتم استدعاؤها عن طريق وضع اسمها في شكل تعليمة حسابية مثل :

X = 5.0* SQRT (A**2 + B**2)

R1 = ROOT (A,B,C)

حيث ROOT يمثل تعبير رياضي لدالة يتم وضعها في أول البرنامج الأصلي أو تمثل برنامج جزئي لدالة تقوم بحساب جذر معادلة من الدرجة الثانية معاملاتها C,B,A مثلا . ويختلف الحال بعض الشيىء في البرامج الجزئية الفرعية حيث يتم استدعاء هذا النوع من البرامج عن طريق التعليمة CALL . والصورة العامة لتلك التعليمة هي :

CALL name (a_1, a_2....., a_n)

حيث name : اسم البرنامج الجزئي الفرعي المطلوب الانتقال اليه .

a_1, a_2....., a_n : متغيرات سبق معرفة قيمها وقد تكون ثوابت أيضاً حقيقية أو صحيحة ، كما قد يمثل أحد تلك المتغيرات أو بعضها النتيجة أو النتائج التي قام البرنامج الجزئي الفرعي بحسابها

(*) قد يكون هذا البرنامج هو البرنامج الأصلي أو برنامج جزئي لدالة أو برنامج جزئي فرعي آخر .

تمهيداً للعودة بتلك النتائج الى الموضع الذي تم منه استدعاء البرنامج الجزئي الفرعي . فعلى سبيل المثال يمكن أن تكون صيغة الاستدعاء للبرنامج الجزئي الفرعي ROOT الذي يقوم بحساب جذري معادلة من الدرجة الثانية معاملاتها هي C,B,A في الصورة :

CALL ROOT (A,B,C ROOT1, ROOT2)

وفي هذه الحالة يجب أن تكون قيم المتغيرات C,B,A معروفة مسبقا ، وقبل أن يتم استدعاء البرنامج الجزئي الفرعي ROOT . ولكن ما هي المتغيرات ROOT1, ROOT2 التي ظهرت في تعليمة الاستدعاء ؟ والاجابة على ذلك هي نتائج الحسابات المطلوبة من البرنامج الجزئي الفرعي أن يقوم بها .

وعند هذه النقطة ، فهناك **اختلافين جوهريين** بين البرامج الجزئية الفرعية وبقية الأنواع الأخرى التي سبق دراستها .

١ – ففي البرامج الجزئية الفرعية يمكن حساب قيم اكثر من متغير (ROOT2, ROOT1 مثلا) والعودة بها الى التعليمة التي تم منها استدعاء البرنامج الجزئي الفرعي ، بينما الأنواع الأخرى سواءً البرامج الجزئية للدوال أو الدوال التي في صورة تعبير رياضي لايمكنها حساب اكثر من قيمة واحدة .

٢ – في البرامج الجزئية الفرعية يتم الحصول على النتائج من اسماء المتغيرات التي حسبت فيها تلك النتائج ، بينما في الأنواع الأخرى فان اسم الدالة هو الذي يتم فيه الاحتفاظ بالنتيجة .

وطبقاً للخاصية الأولى ، فلا نستطيع مثلاً استخدام برنامج جزئي لدالة لحساب مصفوفة ، حيث أن المصفوفة تحوي اكثر من قيمة بينما يمكن عمل ذلك باستخدام البرامج الجزئية الفرعية . وفيما يتعلق بالخاصية الثانية ، فانه نظراً الى أن نتائج حسابات البرامج الجزئية الفرعية يتم تخزينها في متغيرات خاصة بتلك البرامج الجزئية ، الا أنه عند العودة الى البرنامج الأصلي أو الى النقطة التي تم منها الاستدعاء فان تلك النتائج تُفَّرغ في المتغيرات المناظرة لها في تعليمة الاستدعاء CALL فعلى سبيل المثال ، اذا كانت تعليمة الاستدعاء هي :

CALL ROOT (A,B,C, ROOT1, ROOT2)

فهذا لايؤثر بشيء اذا كانت متغيرات البرنامج الجزئي الفرعي كالتالي :

SUBROUTINE ROOT (X,Y,Z, R1, R2)

وفي هذه الحالة ، اذا أريد طباعة قيمتي الجذرين في البرنامج الأصلي فيجب طباعة ROOT2, ROOT1 وليس R_1, R_2 . والمهم في هذا الموضوع أن تكون عدد المتغيرات ونوعها (صحيح أو حقيقي) في تعليمة الاستدعاء يماثل تماماً عدد المتغيرات ونوعها المناظرة لها في البرنامج الجزئي الفرعي ، ولا يهم ان كانت اسماء المتغيرات متشابهة أم لا .

مثـال (١) :

```
        SUBROUTINE ROOT (X,Y,Z,T1, T2)
        R = SQRT (Y*Y-4.0 *X*Z)
        IF (R) 3,2,1
1       S =  2.0*X
        T1 = (-Y+R) / S
        T2 = (-Y-R) / S
        RETURN
2       T1 =  -Y / (2.0*X)
        T2 = T1
        RETURN
3       WRITE (6,4)
4       FORMAT ('IMAGINARY ROOTS')
        T1 = 9999.9
        T2 = T1
        RETURN
        END

C       MAIN PROGRAM
        A = 2.0
        B = 4.0
        C = 1.0
        CALL ROOT (A,B,C, ROOT1, ROOT2)
        WRITE (6,1) ROOT1, ROOT2
1       FORMAT (2F14.4)
        STOP
        END
```

نلاحظ في هذا المثال ما يلي :

١ – في البرنامـج الجزئي الفرعي ، هناك اكثر من تعليمة عودة RETURN الى البرنامـج الأصلي .

٢ – ان هناك تعليمات طباعة يمكن ان يحتويها البرنامـج الجزئي الفرعي .

٣ – ان اسماء المتغيرات التي احتوتها تعليمة الاستدعاء في البرنامـج الأصلي تختلف عن اسماء المتغيرات في البرنامـج الجزئي الفرعي ولكن بشرط أن عدد ونوع تلك المتغيرات واحدة في البرنامجين .

٤ – ان نتائـج البرنامـج الجزئي الفرعي ستحسب في المتغيرات T2, T1 بينما في البرنامـج الأصلي ستستخدم نظيريهما ROOT2, ROOT1 ، وعموماً فقد تتفق أو تختلف اسماء المتغيرات في البرنامجين فلا يهم هذا طالما أننا سبق أن أتفقنا أن كلاً من البرنامجين يعتبر مستقلاً عن الآخر .

٥ – في البرنامـج الأصلي يمكن تغيير تعليمة الاستدعاء الى :

```
CALL ROOT (2.0, 4.0. 1.0, ROOT1, ROOT2)
```

أي أننا نضع قيم المتغيرات بدلاً من اسمائها ولكن لايمكن عمل ذلك في البرنامـج الجزئي الفرعي والا لكان برنامجاً جزئياً فرعياً **خاصـاً** ، وهذا ما لايتفق مع طبيعة استخدام مثل تلك البرامـج الجزئية .

٦ – أن المتغيرات التي تمثل النتائج ، يتم كتابتها بعد بقية المتغيرات .

مثـال (٢) :

لنفرض أن هناك مصفوفتين Y, X يتم حسابهما داخل البرنامـج الأصلي ، ويراد جمعهما اذا كانت جميع عناصرهما موجبة وغير صفرية . وسنستخدم برنامجاً جزئياً لدالة (TEST) لاختبار قيم المصفوفتين ، بينما سنستخدم برنامجاً جزئياً فرعياً (ADDMAT) لجمع هاتين المصفوفتين في حالة ما اذا تحقق الشرط السابق . وسنفترض أن عملية الجمع ستتم في احدى المصفوفتين ولتكن Y .

```
C       MAIN PROGRAM
        DIMENSION X (20, 10), Y(20,10)
           ⋮
        DO 10 I = 1,M
        DO 10 J = 1,N
           ⋮
        X (I,J) = .....
10      Y (I,J) = .....
        CALL ADDMAT (M,N,X,Y,IN)
        IF (IN.EQ.0) GO TO 30
        DO 15 I = 1,M
15      WRITE (6,20) (Y(I,J), J = 1,N)
20      FORMAT (IX, 10 F 12.3)
30      STOP
        END
```

```
C       A SUBROUTINE SUBPROG. TO ADD TWO MATRICES.
        SUBROUTINE ADDMAT (M,N,X,Y,K)
        DIMENSION X(1,1), Y(1,1)
        K = ITEST (M,N,X,Y)
C       TESTING THE POSSIBLITY TO ADD OR NOT.
        IF (K.EQ.Ø) RETURN
        DO 1Ø I=1, M
        DO 1Ø J=1, N
   1Ø   Y(I,J) = Y(I,J) +X(I,J)
        RETURN
        END

C       A FUNCTION SUBPROG. TO TEST THE NUMBERS
        FUNCTION ITEST (M,N,X,Y)
        DIMENSION X(1,1), Y(1,1)
        ITEST = Ø
        DO 5 I = 1,M
        DO 5 J = 1,N
        IF (X(I,J). LT. Ø.Ø) RETURN
        IF (Y(I,J). LT. Ø.Ø) RETURN
    5   CONTINUE
        ITEST = 1
        RETURN
        END
```

نلاحظ في المثال السابق ما يلي :

١ – أن البرنامج الرئيسي يعتمد في تنفيذه على برنامج جزئي فرعي ADDMAT ، كما أن الأخير يحتاج في تنفيذه الى برنامج جزئي لدالة ITEST (أو برنامج جزئي فرعي آخر) .

٢ – أننا نتعامل مع مصفوفات في جميع البرامج وهذا يتطلب ذكر تعليمة DIMENSION لتحديد أبعاد تلك المصفوفات في تلك البرامج ، وفي البرنامجين الجزئيين اكتفينا بتحديد أبعاد المصفوفتين بالمتغيرات X(1,1), Y(1,1) . ولكن في البرنامج الرئيسي لابد من تحديد أبعادهما بالأرقام الفعلية كما أنه في البرنامج الجزئية لايجب أن نتعدى تلك الأرقام (M ≦ 2Ø), (N ≦ 1Ø) .

٣ – بعد الانتهاء من تنفيذ البرنامج الجزئي الفرعي ADDMAT والعودة الى نقطة الاستدعاء في البرنامج الرئيسي ، فان عملية جمع المصفوفتين من عدمها يتوقف على قيمة المتغير K في البرنامج الجزئي الفرعي وهي نفس القيمة للمتغير IN في البرنامج الرئيسي .

٤ – ان المتغيرات المستخدمة في حساب البرامج الجزئية عامة قد تكون جميعها من نوع واحد (حقيقية أو صحيحة) ، وقد تكون خليط منها .

٨-٤-٢ الفروق بين البرامج الجزئية للدالة والبرامج الجزئية الفرعية :

ولعدم التداخل في قواعد وكيفية استخدام البرامج الجزئية للدالة والبرامج الجزئية الفرعية ، سنلخص في الجدول التالي أهم الفروق بينهما .

برامج جزئية	لدالة FUNCTION	فرعية SUBROUTINE
اسم البرنامج الجزئي .	يتكون من ١-٦ حروف وأرقام بحيث يبدأ بحرف ولا يحتوي على علامات خاصة .	يتكون من ١-٦ حروف وأرقام بحيث يبدأ بحرف ولا يحتوي على علامات خاصة .
عدد المتغيرات المستخدمة	متغير أو اكثر .	بدون متغيرات أو بأكثر من متغير .
عدد النتائج التي يمكن العودة بها بعد التنفيذ .	نتيجة واحدة تحملها اسم الدالة .	قد لا يعود بأية نتائج ، وقد يعود بأكثر من نتيجة .
طريقة استدعاء البرنامج الجزئي .	بوضع اسم الدالة ومتغيراتها في صورة تعليمة رياضية .	بإستخدام تعليمة الاستدعاء CALL.
العلاقة بين الاسم والقيمة (أو القيم المحسوبة)	هناك علاقة ، فنوع النتيجة (حقيقية أم صحيحة) يتوقف على اسم الدالة.	ليست هناك علاقة .

وقد يتضح من الجدول السابق ، أن البرامج الجزئية الفرعية تعتبر أكثر مرونة في استعمالها ، ومع ذلك فقد يتطلب الأمر في بعض الأحيان ضرورة استخدام أحد النوعين دون النوع الآخر ، وهذا يتوقف على مهارة وخبرة واضع البرنامج .

٨-٥ تعليمة التعميم : COMMON Statement

مقدمة :

فيما سبق عرفنا أن وسيلة الترابط والمشاركة بين البرنامج الرئيسي والبرنامج الجزئي أو مجموعة البرامج الجزئية التي يتطلب استخدامها مع البرنامج الرئيسي وكذلك بين البرامج الجزئية نفسها تتم عن طريق مجموعة من المتغيرات تظهر في كل من تعليمة الاستدعاء والتي توجد في البرنامج الرئيسي والتعليمة التي تبين اسم البرنامج الجزئي . وعلى سبيل المثال وكما سبق أن أوضحنا فان تعليمة الاستدعاء في البرنامج الرئيسي يمكن أن تكون في الصورة :

```
CALL ADDMAT (M,N,X,Y,K)
```

وفي هذه الحالة فان اسم البرنامج الجزئي قد يكون في الصورة :

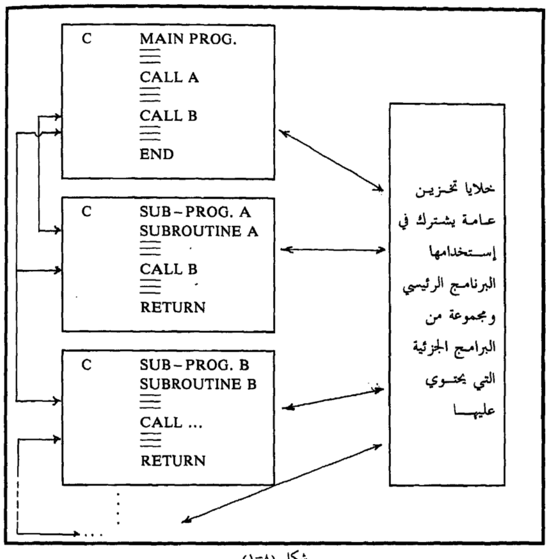
```
SUBROUTINE ADDMAT (NR,NC,A,B,KEY)
```

ولايستلزم على المبرمج أن يستعمل اسماء متغيرات في البرنامج الجزئي تكون مختلفة عن تلك التي استخدمت في تعليمة الاستدعاء ، ولكن يتوجب أن يكون كل متغير في تعليمة الاستدعاء لها نظيرها وفي نفس الترتيب والنوع (حقيقي أو صحيح) في إسم البرنامج الجزئي .

وعندما يحتوي البرنامج الرئيسي على أكثر من برنامج جزئي وكل من تلك البرامج الجزئية يتم استدعاؤها عن طريق مجموعة كبيرة من المتغيرات تظهر في كل من تعليمة الاستدعاء – كما سبق أن بيّنا – وفي اسم البرنامج الجزئي فقد يكون من الأفضل استخدام تعليمة التعميم .

تعليمة التعميم : COMMON Statement

تكمن أهمية تلك التعليمة في تعميم (تخصيص) جزء من خلايا التخزين كي يتم استخدامها عن طريق البرنامج الرئيسي ومجموعة البرامج الجزئية التي يحتوي عليها : بمعنى آخر أن تلك التعليمة تستخدم في تعميم بعض خلايا وحدة التخزين في الحاسب (أنظر الشكل ٨-١) كي يشترك في استخدامها كل من البرنامج الرئيسي وكذلك البرامج الجزئية (كلها أو بعضها) التي يتطلب استدعاؤها من البرنامج الرئيسي . أي أن تلك المنطقة ستستخدم للربط بين البرنامج الرئيسي وبرامجه الجزئية وكذلك بين البرامج الجزئية نفسها وذلك **كبديل** عن استخدام المتغيرات التي تظهر في تعليمة الاستدعاء والتعليمة التي تحتوي عليها اسم البرنامج الجزئي المطلوب استخدامه ، خاصة اذا كان عدد تلك المتغيرات كبيراً . ويجب أن **تسبق** تعليمة التعميم أي تعليمة تنفيذية أخرى في البرنامج الرئيسي أو البرامج الجزئية .



شكل (٨-١)

وهناك صورتين عامتين لتلك التعليمة :

COMMON List أو COMMON/Name/List

حيث :

List : ترمز الى مجموعة المتغيرات (صحيحة أو حقيقية - بسيطة أو ذات أبعاد) التي سيتم تعميم وتخصيص خلايا تخزين لها لاستخدامها عند الضرورة في البرنامج الرئيسي وكذلك البرامج الجزئية .

أمثلـة :

COMMON A,B,C,X (1Ø)

COMMON R (6), J (3), S,T

ويتوقف عدد خلايا التخزين التي سيتم تعميمها على عدد المتغيرات المستخدمة في التعليمة وأبعادها (ان كانت ذات أبعاد) . ويحتوي البرنامـج الرئيسي على تعليمة تعميم واحدة من هذا النوع ، تشتمل على جميع المتغيرات التي ستستخدم في البرامـج الجزئية المختلفة بحيث يوضع في كل برنامـج جزئي تعليمة تعميم واحدة أيضا تشمل نفس المتغيرات بترتيبها وأنواعها التي ظهرت بها في تعليمة التعميم الموجودة في البرنامـج الرئيسي . ويمكن أن يوضع أيضا في كل برنامـج جزئي تعليمة تعميم تحتوي على المتغيرات التي يتطلبها البرنامـج الجزئي فقط دون غيرها من المتغيرات الباقية ، ولكن يجب الحذر هنا حيث أن عملية حجز قيم المتغيرات في تعليمة التعميم يتم بطريقة تتابعية . فمثلاً اذا كانت تعليمة التعميم في البرنامـج الرئيسي في الصورة : COMMON A,D (5),B,C,X (2,3)

وكان هناك برنامجين جزئيين احداهما يحتوي على تعليمة تعميم في الصورة :

COMMON A,B,X (2,3)

والبرنامـج الجزئي الآخر يحتوي على تعليمة التعميم : COMMON A,C,D (5)

فان مجموعة خلايا التخزين الخاصة بتعميم المتغيرات X (2,3), C,B,D (5), A سيتم فيها حجز قيم المتغيرات السابقة بنفس الترتيب الذي كتبت به في تعليمة التعميم في البرنامـج الرئيسي . أي أن خلايا التخزين ستكون في الصورة :

1	2	3	4	5	6	7	8	9	10	1	12	13	14
□	□	□	□	□	□	□	□	□	□	□	□	□	□
A	D_1	D_2	D_3	D_4	D_5	B	C	X_{11}	X_{21}	X_{12}	X_{22}	X_{13}	X_{23}

وعند تنفيذ البرنامـج الجزئي الأول فان قيم المتغيرات التي تحتوي عليها تعليمة التعميم في هذا البرنامـج الجزئي ستكون كالتالي :

المتغير A سيأخذ القيمة المخزنة في الخلية ١ وهي نفس قيمة المتغير A

المتغير B سيأخذ القيمة المخزنة في الخلية ٢ وهي قيمة المتغير D (1)

المتغير X (1,1) سيأخذ القيمة المخزنة في الخلية ٣ وهي قيمة المتغير D (2)

المتغير X (2,1) سيأخذ القيمة المخزنة في الخلية ٤ وهي قيمة المتغير D (3)

المتغير X (1,2) سيأخذ القيمة المخزنة في الخلية ٥ وهي قيمة المتغير D (4)

المتغير X (2,3) سيأخذ القيمة المخزنة في الخلية ٨ وهي قيمة المتغير C

ونفس الوضع سينشأ عند تنفيذ البرنامج الجزئي الثاني حيث أن :

المتغير A سيأخذ القيمة المخزونة في الخلية ١ وهي نفس قيمة المتغير A

المتغير C سيأخذ القيمة المخزنة في الخلية ٢ وهي القيمة التي خصصت للمتغيرين B,D (1)

المتغير D (1) سيأخذ القيمة المخزنة في الخلية ٣ وهي القيمة التي خصصت للمتغيرين X(1,1), D(2)

المتغير D (5) سيأخذ القيمة المخزنة في الخلية ٧ وهي القيمة التي خصصت للمتغيرين X(1,3),B

أي أن بعض خلايا التخزين في تلك المنطقة سيحتوي كل منها على قيمة ستخصص لأكثر من متغير مثل الخلايا ٢ ، ٣ ، ... ، ٨ . ولذا فيجب الحذر عند وضع تعليمات تعميم في البرامج الجزئية تختلف في محتوياتها وترتيب – المتغيرات بها عن تعليمة التعميم التي تحتوي عليها البرنامج الرئيسي . ويتبن من هذا المثال أن :

١ – أي برنامج أساسي أو جزئي يجب الا يحتوي على اكثر من تعليمة تعميم واحدة من هذا النوع .

٢ – ليس من الضروري أن تكون عدد وحدات خلايا التخزين في تعليمات التعميم من هذا النوع متساوية ، ولكن عادة ما يكون عدد وحدات خلايا التخزين في تعليمة التعميم في البرنامج الرئيسي أكبر من أو يساوي عدد وحدات خلايا التخزين لأي تعليمة تعميم موجودة في برنامج جزئي .

٣ – تعليمة التعميم قد تحتوي متغيرات بسيطة أو ذات أبعاد وبالتالي فان تعليمة التعميم يمكن أن تحل محل التعليمة التوضيحية لأبعاد المتغير DIMENSION Statement أي انه اذا أحتوت تعليمة التعميم على متغيرات ذات أبعاد فلا يجب في هذه الحالة ظهور اسماء تلك المتغيرات في التعليمات التوضيحية الأخرى مثل REAL, INTEGER واذا تطلب الأمر توضيح نوع بعض المتغيرات ذات الأبعاد من حيث انها حقيقية أو صحيحة فيمكن عمل ذلك في التعليمة التوضيحية على أن تظهر اسماء تلك المتغيرات في تعليمة التعميم كمتغيرات بسيطة وليست ذات أبعاد . فمثلا يمكن كتابة :

```
INTEGER X (5,5), D (5)
COMMON X,D
```

أو بطريقة أخرى :

```
COMMON IX (5,5), ID (5)
```

مثـال :

باستخدام تعليمة بالتعميم اكتب برنامـج جزئي لجمع مصفوفتين .

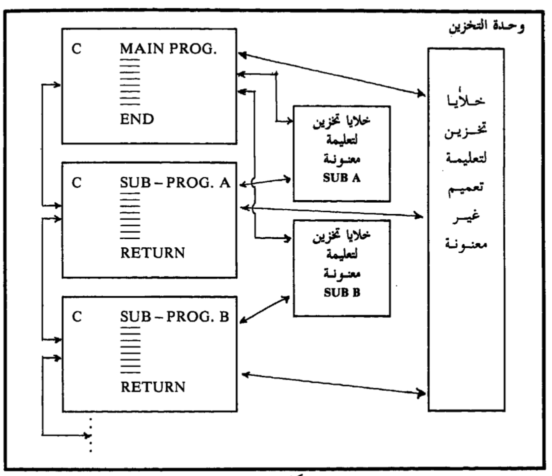
سنفترض أن المصفوفتين هما B,A وأن نتيجة الجمع ستكون في احداهما ولتكن المصفوفة A .

```
C       MAIN PROG
C       NR IS NO. OF ROWS, NC IS NO OF COLUMNS.
        COMMON A (15,15), B (15,15), NR, NC
        READ (5,*) NR, NC
        DO 6 I=1, NR
    6   READ (5,*) (A(I,J), J=1,NC)
        DO 7 I=1, NR
    7   READ (5,*) (B(I,J), J=1, NC)
        CALL MATADD
        DO 8 I=1, NR
    8   WRITE (6,9) (A(I,J), J=1, NC)
    9   FORMAT (1X,10 F12.3)
        STOP
        END
```

```
C       ASUBPROG TO ADD 2 MATRICES OF MAX. ORDER OF (15 × 15)
        COMMON X(15,15), Y(15,15), NR, NC
        DO 3 I=1, NR
        DO 3 J=1, NC
    3   X(I,J) = X(I,J) + Y(I,J)
        RETURN
        END
```

أما الصورة الثانية لتعليمة التعميم فتختلف عن الأولى في أن كل تعليمة تعميم يصحبها اسم يميزها عن بقية تعليمات التعميم التي يحتوي عليها البرنامـج الأصلي والبرامـج الجزئية . وكل تعليمة تعميم من هذا النوع تحتوي على مجموعة من المتغيرات تستخدم في حساب برنامج جزئي ، وقد يشترك متغير أو اكثر في اكثر من تعليمة تعميم . كما أن الاسم الذي يميز تعليمة التعميم عن بقية تعليمات التعميم الأخرى ينطبق عليه نفس الشروط التي تنطبق على اسم أي متغير في لغة الفورتران ، أي يتكون من مجموعة من الحروف والأرقام لايزيد عددها عن ستة وبشرط أن يبدأ بحرف وليس رقما ولا يحتوي على أية إشارات خاصة .

كما يلاحظ أن هذا النوع من تعليمات التعميم يبدأ بكلمة التعميم COMMON ثم يأتي اسم التعليمة بين علامتي قسمة (Slash) ثم تأتي مجموعة المتغيرات التي تختص بها تلك التعليمة . كما أن لكل تعليمة تعميم من هذا النوع مجموعة من خلايا التخزين تختص بتلك التعليمة دون غيرها (أنظر الشكل ٨-٢) ، بحيث لاتتداخل خلايا تخزين تعليمة ما مع خلايا تخزين - تعليمة أخرى . ولذا فيمكن أن نطلق على هذا النوع من تعليمات التعميم اسم **تعلميات التعميم المعنونة**
Labeled COMMON Statement



شكل (٨-٢)

ومن الأفضل وضع المتغيرات التي لا تأخذ قيماً عددية مثل المتغيرات المنطقية Logical Variables أي التي تأخذ احدى القيمتين TRUE أو FALSE والمتغيرات الأخرى التي تمثل قيمها بحروف وليس بأرقام عددية أن توضع مثل تلك المتغيرات في تعليمات تعميم معنونة منفصلة عن تعليمات التعميم الأخرى ، أي لاتحتوي تعليمة التعميم على خليط من - المتغيرات ذات القيم العددية وغير العددية .

والاستخدام الأمثل لمثل هذا النوع من تعليمات التعميم يكون عندما يتطلب الأمر وجود مجموعة من البرامـج الجزئية بجانب البرنامـج الأصلي ، وكل برنامـج جزئي يتطلب لحسابه مجموعة من المتغيرات قد تختلف جزئياً أو كلياً عن مجموعة المتغيرات اللازمة لحساب البرنامـج الجزئي الآخر . وفي هذا النوع من تعليمات التعميم **يجب** أن يكون عدد وحدات خلايا التخزين في تعليمة التعميم المعنونة والتي توجد في البرنامـج الأصلي **مساوية تماماً** لعدد وحدات خلايا التخزين في تعليمة التعميم المعنونة والموجودة في البرنامـج الجزئي . وقد يكون هذا أحد الفروق الأساسية بين تعليمات التعميم العادية وتعليمات التعميم المعنونة . كما يمكن جمع تعليمتي تعميم أو أكثر من هذا النوع في تعليمة تعميم واحدة .

فعلى سبيل المثال ، بدلاً من كتابة :

COMMON / A / VAR1, VAR2, MAT (3,3)
COMMON / B / VAR3, VAR4, VAR5, ARRARY (8)

كلاً على حدة ، يمكن دمجهما في تعليمة تعميم واحدة في الصورة :

COMMON /A/ VAR1, VAR2, MAT (3,3) /B/ VAR3, VAR4, VAR5, ARRAY (8)

٨-٦ تعليمة التكافؤ : EQUIVALENCE Statement

مقدمـة :

من المعروف أن لكل متغير خلية (عنوانا في وحدة التخزين) أو مجموعة من خلايا التخزين تنسب الى هذا المتغير وتحتفظ بقيمته فيه كي يستخدم كلما ورد اسم ذلك المتغير في البرنامـج . وقد يكون من الغريب أن تخصص خلية تخزين لأكثر من متغير في نفس البرنامـج ، ولنا أن نتوقع ان امكن تنفيذ ذلك – مدى ما ستوفره من خلايا تخزين قد تكون ذات فائدة ، خاصة اذا كان البرنامـج كبيرا ويحتوي على مجموعة كبيرة من المتغيرات والتي قد يستعمل بعضها في اجزاء معينة من البرنامـج دون الحاجة اليها في مواضع أخرى من نفس البرنامـج ، مما يجعل مثل تلك المتغيرات تحتل مساحات في وحدة التخزين قد تكون في مسيس الحاجة اليها خاصة اذا كانت سعة وحدة التخزين في الحاسب محدودة . ولكن تعليمة التكافؤ « في لغة الفورتران » تسمح بتنفيذ ذلك بشرط الا يتواجد متغيرين أو مجموعة من المتغيرات تتقاسم خلية تخزين واحدة على يمين علامة التساوي في تعليمة تنفيذية واحدة في البرنامـج . كما أن هناك نوع من الارتباط بين تعليمة التكافؤ وتعليمة التعميم التي سبق شرحها ، سنحاول توضيحه فيما بعد .

٨-٦-١ الصورة العامة لتعليمة التكافؤ :

تأخذ تعليمة التكافؤ الصورة العامة التالية :

```
EQUIVALENCE (V1, V2, ....., Vm)
```

حيث $V_1, V_2, \ldots\ldots, V_m$ مجموعة متغيرات تحتل مكاناً واحداً في وحدة التخزين . وقد تكون بعض تلك المتغيرات بسيطاً وبعضها الآخر تمثل عناصر في مصفوفة . ويلاحظ أن مجموعة المتغيرات تلك توضع بين قوسين .

أمثلـة :

```
EQUIVALENCE (A(3),X)
EQUIVALENCE (R,S,K,V)
```

ويمكن دمج أكثر من تعليمة تكافؤ في تعليمة واحدة . فعلى سبيل المثال يمكن كتابة تعليمتي التكافؤ السابقتين في تعليمة واحدة كالتالي :

```
EQUIVALENCE (A(3),X), (R,S,K,V)
```

وهناك بعض الشروط في استخدام تعليمة التكافؤ في أي برنامج ، منها :

١ - يجب أن تسبق تعليمة التكافؤ أي تعليمة تنفيذية في البرنامج وكذلك الدوال ذات التعبير الرياضي ، كما أنها يجب الا تسبق التعليمات التوضيحية مثل REAL, INTEGER, DIMENSION

٢ - كل مجموعة متغيرات يراد أن يكون لها نفس خلية التخزين ، يجب أن تحتوي على متغيرين على الأقل .

٣ - لايشترط أن تكون مجموعة المتغيرات المتكافئة من نوع واحد ، أي جميعها حقيقية أو صحيحة ولكن يمكن أن يكون بعضها حقيقي والآخر صحيح طالما أنه لايوجد أكثر من متغير واحد من تلك المتغيرات على يمين علامة التساوي من تعليمة تنفيذية . فمثـلاً :

```
EQUIVALENCE (DEGFHT, K)
READ (3,*) DEGFHT
DEGCNT = DEGFHT - 32.
   ⋮        ⋮
K = I + J
```

فالمتغير DEGFHT (فهرنهيتية) لن يستخدم مرة أخرى في البرنامج بعد تحويله الى DEGCNT (درجات مئوية) ، وطالما أن تلك التعليمة لاتحتوي على المتغيرين K, DEGFHT معا .

٤ – في تعليمة التكافؤ بين متغير ومصفوفة يوضع عنصر من المصفوفة كأحد المتغيرات ولا توضع المصفوفة كلها ، الا اذا كان التكافؤ بين مصفوفتين في جميع عناصرهما . فمثلاً اذا كانت R تمثل متغير ما وكانت X تمثل مصفوفة تحتوي على خمس قيم ، فلا يمكن كتابة تعليمة التكافؤ التالية :

EQUIVALENCE (X, R)

ولكن يمكن أن يحتل المتغير R وأحد عناصر المصفوفة X خلية ما في وحدة التخزين .

أي يمكن كتابة : EQUIVALENCE (X(3), R)

والتي تعني أن المتغير R يشترك مع العنصر الثالث من المصفوفة X في نفس خلية التخزين . ويمكن تخيل ذلك في وحدة التخزين كالتالي :

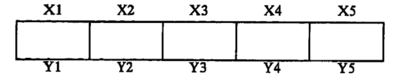
X_1 X_2 X_3 X_4 X_5

R

بينا يمكن كتابة تعليمة التكافؤ بين عناصر المصفوفة X جميعها وعناصر مصفوفة أخرى Y ذات خمس قيم أيضاً بإستخدام التعليمة :

EQUIVALENCE (X,Y)

وفي هذه الحالة ستكون خلايا التخزين المشتركة في الصورة :



ونفس خلايا التخزين المشتركة السابقة يمكن تحقيقها باستخدام تعليمة مثل :

EQUIVALENCE (X(1), Y(1))

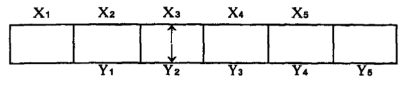
أو EQUIVALENCE (X(2), Y(2))

وهكذا ، بحيث يكون الدليل العددي للمتغير X يتطابق مع الدليل العددي للمتغير Y وفي جميع الحالات السابقة يعتبر التكافؤ بين المصفوفتين كاملا ، أي أن كل خلية من الخلايا الخمس تحجز لمتغيرين .

ولكن اذا كانت تعليمة التكافؤ هي :

EQUIVALENCE (X(3), Y(2))

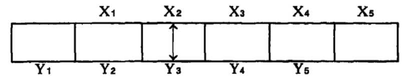
فإن مجموعة خلايا التخزين ستكون في الصورة :



واذا كانت تعليمة التكافؤ في الصورة :

EQUIVALENCE (X(2), Y(3))

فستكون خلايا التخزين في الصورة :



واذا افترضنا أن احدى المصفوفتين ولتكن X ذات بعدين (2,4) X ، فان تعليمة تكافؤ مثل :
EQUIVALENCE (Y(2), X(2,2))

تكون خلايا التخزين في تلك الحالة في الصورة :

X_{11} X_{21} X_{12} X_{22} X_{13} X_{23} X_{14} X_{24}

Y_1 Y_2 Y_3 Y_4 Y_5

واذا كانت المصفوفة Y تحتوي على ٥٠ قيمة مثلا ، فإن نفس التعليمة السابقة يمكن تمثيلها كالتالي :

X_{11} X_{21} X_{12} X_{22} X_{13} X_{23} X_{14} X_{24}

Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7

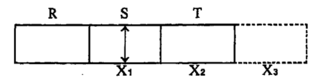
٨-٦-٢ الارتباط بين تعليمتي التعميم والتكافؤ :

من الممكن ظهور اسم متغير بسيط أو ذو أبعاد في كل من تعليمتي التعميم والتكافؤ في نفس البرنامج ، ولكن هناك بعض القيود التي يجب أخذها في الاعتبار حتى نتجنب المشاكل التي قد تنجم نتيجة لتواجد المتغير في التعليمتين مما قد ينتج عنه زيادة في منطقة التعميم قد يصرّح بها أو لايصرح بها .

فإذا أفترضنا الجزء التالي من البرنامج والذي يحتوي على تعليمة تعميم غير مُعنْونة ،

```
REAL X (3)
COMMON R,S,T
EQUIVALENCE (X(1), S)
```

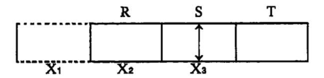
فإن خلايا التخزين لهاتين التعليمتين ستكون كالتالي :



أي أننا سنضطر الى اضافة بعض خلايا التخزين (نحو اليمين) في منطقة التعميم ، وهذا لاينتج عنه مشاكل . ولكن اذا كانت تعليمة التكافؤ في جزء البرنامج السابق في الصورة :

EQUIVALENCE (X(3), S)

فإن خلايا التخزين ستكون كالتالي :



وهذا يعني أننا نريد من الحاسب حجز خلية اضافية تسبق منطقة التعميم ، كي يوضع فيها العنصر X(1) وهذا غير مسموح به نظراً لأن عملية حجز قيم المتغيرات تتم بطريقة تتابعية مما يعني أن هذه الخلية الاضافية يكون الحاسب قد سبق حجزها لمتغير آخر خارج منطقة التعميم . وببساطة :

يمكن زيادة مساحة منطقة التعميم بالاتجاه نحو اليمين ولا يمكن ذلك بالاتجاه نحو اليسار .

– كذلك غير مسموح بكتابة تعليمة تكافؤ لمتغيرين (أو أكثر) بسيطين أو ذوي أبعاد اذا كان هذين المتغيرين سبق ذكرهما في تعلمية تعميم . فمثلاً من غير المنطقي كتابة الجزء التالي من برنامج .

```
COMMON U, V, W, X (3)
EQUIVALENCE (V, X(2))
```

حيث أنه في هذه الحالة فإن تعليمة التعميم ستخصص خلية تخزين لكل متغير أو عنصر في مصفوفة ، وبالتالي فان تعليمة التكافؤ لن يمكن تحقيقها حيث أن خلايا التخزين ستكون في الصورة :

U	V	W	X_1	X_2	X_3

والتي يتضح منها أن أي متغيرين من متغيرات تعليمة التعميم لن يمكن جمعهما في خلية تخزين واحدة .

– بالنسبة لتعليمات التعميم المعنونة والتي توجد في البرنامج الرئيسي واحد أو بعض البرامج الجزئية ، يجب أن يكون عدد خلايا التخزين متساوياً في تعليمات التعميم التي تحمل نفس الاسم . وبالتالي فإن إضافة أي خلايا تخزين أخرى لتلك التعليمة يجب أن يتم أيضاً في باقي تعليمات التعميم في البرامج الجزئية الأخرى .

« تمـــارين »

١ – بين أي من التعليمات التالية يعتبر مسموحاً به (كل تعليمة تعتبر مستقلة عن غيرها) :

```
COMMON R, S, T(3,3), I,J
COMMON (X(2,2), ARRAY)
COMMON A, 6.2, B
COMMON X, Y, Z/LITS/A, B, C
COMMON/XYZ/X, Y, Z
EQUIVALENCE (X(5,5), Y(3,3))
EQUIVALENCE (X,Y,Z(2)), (B,L)
EQUIVALENCE (X(1), VAR, X(2), L)
EQUIVALENCE (A, 6.0, B)
```

٢ – أذكر قيمة المتغيرات الموجودة في كل من البرامج الجزئية التالية :

```
SUBROUTINE UN
DIMENSION A(3), C(2)
COMMON /XXX/ A,B /YYY/ C
A(2) = 1.0
C(1) = 1.0
RETURN
END
SUBROUTINE DEUX
DIMENSION B(3), C,D(2)
COMMON/XXX/C,B/YYY/D
B(2) = 2.0
C = 2.0
RETURN
END
```

```
SUBROUTINE TROIS
REAL X(4)
COMMON/XXX/X/YYY/A,B
X(1) = 2.0
X(2) = 2.0
X(3) = 2.0
X(4) = 2.0
A = 2.0
B = 2.0
RETURN
END
```

٣ – في البرنامج التالي ، يتم استدعاء البرامج الجزئية السابقة . أكتب قيم المتغيرات في تعليمة الطباعة .

```
      REAL X, Y, Z, R, S, T
      COMMON / XXX / X,Y,Z,R / YYY / S,T
      CALL TROIS
      CALL DEUX
      CALL UN
      WRITE (7,1) X,Y,Z,R,S,T,
1     FORMAT (1X, 6E12.3)
      STOP
      END
```

٤ – بين اي من مجموعة التعليمات « أ » أو «ب» يعتبر مسموحا به .

المجموعة « أ »

```
      DIMENSION A(3),B(4),C(2)
      COMMON / ZZ / A
      EQUIVALENCE(A(2),B(2),C(1))
```

المجموعة «ب»

```
      DIMENSION A (3), B(3)
      COMMON / ZZ / A
      EQUIVALENCE (A(1),B((2))
```

« تمارين عامة »

١ – أعد صياغة التعبيرات التالية بلغة الفورتران :

$$Radius = \sqrt{\frac{a + b^h - 10^6}{(b^2 + 3a)}} ; Area = \frac{314\ R^3}{100\ (a+b)}$$

$$b = \frac{-(x.y)^3 + (x.z)^2 - (y.z)}{x.y.z} ; Area = \sqrt{S(S-a)\ (S-b)\ (S-c)}$$

$$Diameter = \frac{x^z y^2 - Z_2}{xy - Z} - 1; \quad High = \frac{1}{2}(x-y)^2 - \frac{1}{3}(y-z)^3 + \frac{1}{2}(z-x)^4$$

$$S = a \div b + \frac{b\text{-}c}{d}\left(\frac{e^5}{f}\right) ; D = \frac{pb^3}{L^2}(3a + b)$$

$$ArC = 2\sqrt{y^2 + \frac{4x^3}{3}} ; R = \frac{x}{y\text{-}z}(3x^2 - 6x)$$

٢ – اذا كانت :

X=2, Y=3, Z=4, J=2, K=2, KK=7, L=-3

وضح أسبقية تنفيذ مكونات التعليمات التالية ، ثم احسب قيمتها :

a) S= (-(X*Y) **3+ (X*Z) **2- Y*Z) / (X*Y*Z)

b) I= (J*(K-KK) / (9+L)) ; d) III=J*(K-KK) / (9+L)

c) II= (J*(K-KK)) / (9+L) ; e) IV= J* ((K-KK) / (9+L))

٣ – اذا كانت :

A=2. B=1., C=1., D=3., E=2., F=1., G= -1.

أوجد القيمة التي سيختزنها الحاسب في وحدة التخزين للمتغيرات التالية مع توضيح كيفية خطوات الحساب :

Q = ((A/B*D) **2/D) **E

R = A*B/C*D/E*F

J = A + (B+C -(D*E+F))

T = A + B/C + D**E*F-G

٤ – اذا كانت : $A = 1., B = 2., C = 1., D = 2., E = 3., F = -1., G = 1.$

أوجد القيمة التي سيختزنها الحاسب في وحدة التخزين للمتغيرات التالية ، مع توضيح كيفية خطوات الحساب :

```
Q =  A*((B/C*D)/E) *F-G
J =  ((A+B)/C) + D**E*(F-G)
S =  A*((B+C) - (D*E) + F)
T =  A/((D*D-G) **2/D) **E
LU = Ø.5* (A-B) **2-(B-C) **3/3.+Ø.5*((C-B)/D) **2
```

٥ – أعد كتابة التعليمات التالية في صورتها الصحيحة (كل تعليمة مستقلة عن الأخرى) .

```
      DIMENSION X(1ØØ), Y(N), Z(M,1Ø)
      Z = Z + A*J
      W(I) = W**2 + W(I,J)
      Y = 2Y/-A**Ø.5
      A(I) = I+B-J/R
      IF (X-1ØØ) - 1Ø,Ø,+1ØØ
      GO TO K
      READ ((7, 1Ø) A(I,J), I=1,K, J=1,L)
      GO TO I,(1Ø,2Ø,3Ø,1Ø)
      DO 1Ø I=1,M,N,
```

٦ – في البرنامج التالي بعض الأخطاء ، أعد كتابته بصورة صحيحة :

```
51    A+7D = C
      R=2.*L + C + B
      READ = (7,I) A,B,L,D.
1     FORNAT (F8,3-2F6.2,13)
5     GO TO 5
      IF (L - C) 7,5
7     WARITE (7.51) A,B,R-C
      STOB
      END
```

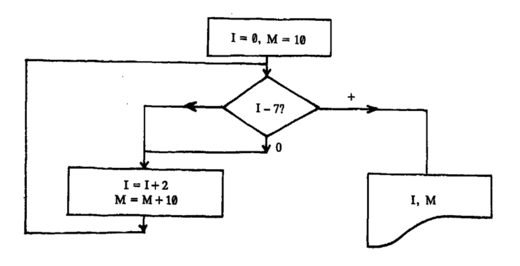
٧ - في البرنامـج التالي توجد عدة اخطاء في لغة الفورتران ، أعد كتابة البرنامـج بعد تصحيحها .

```
    READ (7,3.) AI,IA,A.B.
3   FORHAT (2F7,2 , F8.3/)
    IF (AB+AI-IA) 2,1
    DO 5 I=1, AI
2   IS = I + AI
    I = I+IS
5   CONTINUE
    GO TO (2,1,3), AI
1   IF (IS. NE. AB) GO TO 2
    A*I=AI
    WARITE (7,7)A*I,IA,AI
    STOB
    END
```

٨ - في البرنامـج التالي بعض الأخطاء . أعد كتابته بصورة صحيحة .

```
    READ (7,8) A,7B, KA, AK
8   FORHAT (F6, 3, 15, 2I3)
    A + AK = KA
    WRIT (7,7) (R,B7,KA)
    3 FOURMAT I5, F9. 12,6I
    STOP
    END
```

٩ - بافتراض مخطط التدفق التالي :



ماهي القيم النهائية للمتغيرات I ، M ؟

١٠- أعطيت مجموعة من البيانات عن عينة من طلبة وطالبات الجامعة عددها N بحيث أن كل بيان فيها يحتوي على :

١ - مستوى الطالب أو الطالبة حيث يوجد ٤ مستويات هي :

(LEVEL 1, LEVEL 2, LEVEL 3, LEVEL 4)

٢ - نوع الجنس ISEX التابع له هذا البيان (=1= اذا كان طالبا ، =2= اذا كانت طالبة)

أرسم مخطط تدفق التعليمات لحساب :

١ - مجموع الطلبة والطالبات في كل مستوى من المستويات الأربعة في هذه العينة .

٢ - مجموع الطلبة في هذه العينة . ٣ - مجموع الطالبات في هذه العينة .

١١- باعت شركة الطيران السعودية هذا الشهر عدد N من التذاكر لركابها المتجهين لبلدة ما ، بعضها للأطفال الذين يقل أو يساوي عمرهم ١٠ سنوات والباقي لمن يزيد عن ذلك . فإذا كان سعر تذكرة الطفل هي K ريال ، سعر التذكرة للكبار هي L ريال ، أرسم مخطط تدفق تعليمات لحساب :

١ - حصيلة الشركة من تذاكر لأطفال . ٢ - حصيلة الشركة من تذاكر الكبار .

١٢- مصنع يعمل به N عامل وعاملة ، وكل منهم له حفيظة نفوس مبين فيها :

- النوع (١ اذا كان ذكرا ، ٢ اذا كانت انثى) .
- السن (من ١٨ سنة الى ٥٥ سنة) .
- محل الميلاد(١ اذا كان في شمال المملكة ، ٢ اذا كان في شرق المملكة ، ٣ اذا كان في غرب المملكة ، ٤ اذا كان من مواليد جنوب المملكة) .

ارسم مخطط تدفق التعليمات لحساب :

١ - عدد العاملين - عدد العاملات .

٢ - عدد العاملين والعاملات الذين يقل عمرهم عن ٣٠ سنة .

٣ - عدد العاملين والعاملات الذين ولدوا في كل منطقة من مناطق المملكة الأربعة .

١٣- تقوم شركة الكهرباء باستخراج فواتير شهرية مدون عليها رقم الاشتراك NFAC وقراءة العداد السابقة LREAD والقراءة الحالية NREAD وكذلك قيمة الاستهلاك . وَيتم حساب

قيمة الاستهلاك بطرح قراءة العداد السابقة من القراءة الحالية . فاذا كان الفرق (عدد الكيلوات - المستهلكة) ID :

- حتى ١٠٠ كيلو يحسب الاستهلاك كالتالي : CONS1 = ID0 X 0.08 + 10
- مايزيد عن ١٠٠ وحتى ٤٠٠ كيلو وليكن عددها ID1 : CONS1 = ID1 X 0.15 + 25:
- ما يتبقى فوق ٤٠٠ كيلو وات وليكن ID2 : CONS2 = ID2 X 0.2 + 40:

(أي أن : ID = ID0 + ID1 + ID2 ، قيمة الاستهلاك = CONS2 + CONS1 + CONS

ارسم مخطط تدفق تعليمات لتلك البيانات مبتدئاً بقراءة البيانات اللازمة عن كل مشترك (أو مستهلك) ومنتهياً بكتابة الحسابات والبيانات الضرورية لكل فاتورة .

١٤- عدد العمال في مؤسسة ما هو N عامل ، بعضهم يعمل ٣٥ ساعة أسبوعياً أو أقل ويتقاضى كل من هؤلاء ٢٠ ريالا عن كل ساعة . وآخرين يعملون أكثر من ذلك ويتقاضون ٢٥ ريالا عن كل ساعة اضافية . ارسم مخطط تدفق تعليمات لحساب :

١ - عدد أفراد كل فئة . ٢ - مجموع ما تدفعه الشركة لكل فئة .

١٥- أعطيت مجموعة من الأعداد عددها N بعضها زوجي (تقبل القسمة على ٢ بدون باق) ، وبعضها الأخر فردي (نتيجة القسمة تحتوي على باق) . كما أن بعضها موجب والآخر سالبا .

ارسم مخطط تدفق تعليمات (مع توضيح كيفية معرفة العدد الزوجي من الفردي) لحساب :

١ - مجموع الأعداد الفردية SODD
٢ - مجموع الأعداد الزوجية SEVEN
٣ - مجموع الأعداد السالبة SNEG
٤ - مجموع الأعداد الموجبة SPOS

١٦- في أحد المصانع يتم حساب صافي دخل NETPAY كل عامل طبقاً للمعادلة التالية :

NETPAY = BP + OVT − ENCT − OTDED

حيث :

BP	تمثل الدخل الأساسي	Base Pay
OVT	الأجر الاضافي	Over Time
ENCT	ضريبة الدخل	Income Tax
OTDED	خصومات أخرى	Other Deduction

فاذا كان الدخل الأساسي BP الشهري لكل عامل يعتمد على عدد السنوات NY التي قضاها في المصنع بحيث أن :

BP = 800* NY	عندما	(NY < 5)
BP = 1000* NY	»	(5≦NY<8)
BP = 1500* NY	»	(NY > 8)

ارسم مخطط تدفق تعليمات للخطوات التي سيتم بها حساب صافي دخل العاملين بذلك المصنع والذين يبلغ عددهم N ، مبتدئاً بقراءة البيانات اللازمة لكل عامل ثم كتابة بيانات العامل وصافي دخله .

١٧- عند اعطاء البرنامج التالي الى الحاسب ، ظهرت بعض الأخطاء في لغة الفورتران . أعد كتابتها في الصورة الصحيحة .

```
      DIMENSION X (100.)
      READ (7.1) N
 1    FORMATE (20 I3)
      READ (7.40) X(L), I=1, L)
      S = 0
      K = 0.
 3    K = K+1.
      IF (K. GT N.) GO TO 81
      S = S + (KX)
      GO TO 3.
 18   R = S/N.
      WRITE (7,50) N, R.
 50   FOURMAT (1X, THE NO. OF OBSERVATIONS IS. F14.8,5X,
      13 HVALUE OF R = I4.
      STOP
 40   FORMAT (3(F6,2.2X)
      END
```

بعد اعادة كتابة البرنامج في صورته الصحيحة ، فان قيم X التي يتطلب قراءتها كانت كالتالي :

	1	2	3	4	5
X	13.22	31.40	12.08	21.18	22.12

أدخل تلك البيانات الى الحاسب بصيغة الادخال المعطاه في البرنامج أعلاه . وأكتب النتائج التي سيكتبها الحاسب طبقاً لصيغة الطباعة المعطاه في البرنامج .

١٨- عند اعطاء البرنامج التالي الى الحاسب ، ظهرت بعض الأخطاء في لغة يالفورتران . أعد كتابتها في الصورة الصحيحة .

```
        DIMENSION X1 (1Ø). X2 (1Ø), X3 (1Ø.)
        READ (7,L) M
  6     FORMAT (3ØI5)
        READ (7,1Ø) XL (J), K=1,M),(X2(I,I=1,M)
        L = Ø
  8     L = L + 1.
        IF ((L). GT. (M)) GO TØ 1Ø
        X3 (L) = X1. (L) + X5 (L)
        GO TO 8.
 1Ø     FORNAT (3(F4.1,2X)
 1ØØ    WARITE (7.2) X3 (I), I=1-M
  2     FORMAT (IX, THE 16. TH RESULT IS,/ 5X,3F6.1))
        STOP
        END
```

بعد اعادة كتابة البرنامج في صورته الصحيحة ، كانت قيم X2,X1 المطلوب اجراء العمليات الحسابية عليهما كالتالي :

	1	2	3	4	5
X1	−5.1	1.9	0.6	12.3	8.2
X2	6.1	8.2	−5.6	7.7	−3.2

أدخل تلك البيانات الى الحاسب بالصيغة المعطاه في البرنامج ، وأكتب النتائج التي ستحصل عليها من الحاسب طبقاً لصيغة الطباعة المعطاه في البرنامج .

١٩- في التمرين (١٦) أكتب البرنامج المناسب مستخدماً الجمل الشارحة المبينة للبيانات الداخلة حيث :

المتغير NY يأخذ الصورة xx

المتغير OVT يأخذ الصورة xx.xx

المتغير ENCT يأخذ الصورة xx.xxx

المتغير OTDED يأخذ الصورة xxx.xxx

٢٠- في التمرين (١٣) أكتب البرنامج المناسب مستخدماً تعليمات الادخال والطباعة المناسبة والجمل المناسبة اذا كان :

المتغير	LREAD	يأخذ الصورة	xxxxxx	
والمتغير	NREAD	يأخذ الصورة	xxxxxx	
والمتغير	NFAC	يأخذ الصورة	xxxxxx a1 a2	(حيث a2, a1 تمثل
والمتغير	CONS	يأخذ الصورة	xxxxx.xxx	حروفا وليس أرقاما)

٢١- اكتب برنامج لحساب :

$$SUM = \sum_{i=50}^{100} \frac{x_i^2}{i} - \sum_{i=1}^{20} x_i$$

٢٢- أكتب برنامج لقراءة مصفوفة غير خطية X عن طريق صفوفها (صف في كل سطر) ثم اعادة كتابتها بحيث يكتب العمود الأول في السطر الأول ، العمود الثاني في السطر الثاني وهكذا .

٢٣- لديك مجموعة من الأعداد : $X_1, X_2, X_3, \ldots\ldots X_N$

أكتب برنامج لحساب الانحراف المعياري لها (STDEV) حيث :

$$VAR\ iance = \sum_{i=1}^{N} (x_i - \bar{x})^2/N \ ; \ \bar{x} = \sum_{i=1}^{N} x_i/N \ ;$$

$$Standard\ DEV = \sqrt{VAR\ iance}$$

٢٤- أكتب برنامج لحساب الانحراف المعياري STDV لمجموعة من القيم X_i عددها N . مع العلم بأن الانحراف المعياري يمكن حسابه من المعادلة :

$$R^2 = \frac{\sum X_i^2}{N} - \left(\frac{\sum X_i}{N}\right)^2; \quad STDV = \sqrt{R^2}$$

٢٥- مجموعتان من الأعداد عدد كل منها n ، سبق تخزينهما في مصفوفتين خطيتين Y , X . أكتب برنامج .

(أ) لحساب $\sum X_i \ \sum Y_i \ \sum X_i^K Y_i$ وذلك باستخدام حلقات التنفيذ المتكررة

(ب) حساب ماسبق في الخطوة أ بدون استخدام حلقات التنفيذ المتكررة .
(K يمثل عدد صحيح موجب)

٢٦- بافتراض أن لديك مجموعة من الأعداد الموجبة عددها N ، تم حسابها وتخزينها في الحاسب في مصفوفة خطية X . أكتب برنامج :
(أ) لاستخراج أكبر تلك الأعداد وموقعه بين مجموعة تلك الأعداد .
(ب) لاستخراج أصغر تلك الأعداد وموقعه بين مجموعة تلك الأعداد .
(جـ) لاعادة ترتيب تلك الأعداد ترتيباً تنازلياً أي أن :

$$x(1) \geq x(2) \geq x(3) \ldots\ldots \geq x(N)$$

٢٧- في التخزين السابق ، وقبل اعادة ترتيب الأعداد تنازلياً ، أكتب برنامج لاستخراج أكبر خمسة أعداد في المصفوفة الخطية X ومواقعها .

٢٨- بافتراض أن A_{mxn} تمثل مصفوفة عدد صفوفها M وعدد أعمدتها N وتحتوي على مجموعة من الأعداد الموجبة ، أكتب برنامج :
١ - لقراءة تلك المصفوفة عن طريق أعمدتها .
٢ - لاستخراج أكبر عنصر في تلك المصفوفة وموقعه (في أي صف وأي عمود) .
٣ - لاستخراج أصغر عنصر في تلك المصفوفة وموقعه (في أيّ صف وأي عمود) .
٤ - لاعادة ترتيب أعداد تلك المصفوفة ترتيباً تصاعدياً ، أي أن :

$$a_{1,1} \leq a_{1,2} \ldots \leq a_{1N} \leq a_{21} \leq a_{22} \ldots \leq a_{2N} \leq a_{31} \ldots \leq a_{MN}$$

٢٩- المصفوفة $A_{M.M}$ تحتوي على مجموعة من الأعداد الموجبة سبق تخزينها في الحاسب ، ويراد كتابة برنامج لايجاد أكبر عنصر في كل صف كي يجمعه على أكبر عنصر في كل عمود مناظر لذلك الصف .

أي أن المطلوب ايجاد مجموع :
أكبر عنصر في الصف الأول + أكبر عنصر في العمود الأول .
أكبر عنصر في الصف الثاني + أكبر عنصر في العمود الثاني .
وهكذا .

٣٠- عند ضرب مصفوفة A تحتوي على M من المصفوف N من الأعمدة في مصفوفة خطية X تحتوي على N من القيم ، فان ذلك يتم بضرب عناصر كل صف من المصفوفة A في عناصر المصفوفة الخطية وجمع ناتج حواصل الضرب . ولذا فان النتيجة ستكون مصفوفة خطية أخرى ولتكن Y تحتوي على M من القيم حيث :

$$Y_i = \sum_{j=1}^{N} a_{ij} * x_j$$

أكتب برنامج يحقق ذلك .

٣١- في التمرين السابق بإفتراض أن X هي مصفوفة أخرى عدد صفوفها N وعدد أعمدتها L ، فإن عملية ضرب المصفوفتين تتم بضرب كل صف من المصفوفة A في الأعمدة المختلفة للمصفوفة X حسب الطريقة المبينة في التمرين السابق . وبالتالي فإن مصفوفة حاصل الضرب Z (مثلا) ستحتوي على M من الصفوف ، L من الأعمدة ويكون العنصر العام Z_{ik} فيها هو :

$$Z_{ik} = \sum_{i=1}^{M} \sum_{k=1}^{L} \sum_{j=1}^{N} a_{ij} * x_{jk}$$

٣٢- أكتب برنامج لايجاد مقلوب Transpose مصفوفة عدد صفوفها M وعدد أعمدتها N . (يمكن الحصول على مقلوب مصفوفة بجعل عناصر الصف i في المصفوفة الأصلية تمثل عناصر العمود رقم i في مقلوبها) .

٣٣- أكتب برنامج لحساب عدد الأرقام الفردية وعدد الأرقام الزوجية المحصورة بين ١، ٩٩٩ .

٣٤- في برنامج ما يراد طباعة ١٠٠٠ سطر من النتائج . أكتب التعليمات اللازمة لطباعة تلك الأسطر بحيث يكون هناك سطراً خالياً بعد طباعة كل خمسة أسطر .

٣٥- أكتب برنامج لقراءة أي ثلاث قيم تمثل ثلاث أطوال ليبين ان كانت تلك الأطوال تمثل مثلثاً أم لا . وفي حالة ما اذا كانت تلك الأطوال تمثل مثلثاً ، اختبر ما اذا كان مثلثاً قائم الزاوية وذلك بطبع قيم الأطوال الثلاثة ووضع إشارة أو علامة أمام كل ثلاث قيم تمثل مثلثاً قائم الزاوية .

٣٦- نصف قطر الدائرة التي يمكن رسمها داخل مثلث يعطي من القانون :

$$R = \sqrt{\frac{S(S-a)(S-b)(S-c)}{S}} \; ; \; S = \frac{1}{2}(a + b + c)$$

حيث c, b, a هي أطوال أضلاع المثلث . استخدم التمرين السابق كبرنامج جزئي فرعي لاختبار ما اذا كانت الأطوال c, b, a تكون مثلثاً أم لا .

٣٧- زمن الذبذبة بالثواني T لبندول طوله L بوصة يعطي من العلاقة

$T = \pi \sqrt{\frac{L}{g}}$ حيث g

تمثل عجلة الجاذبية الأرضية بالبوصة لكل ثانية مربعة . أكتب برنامج لحساب زمن الذبذبة لبندول يتغير طوله من ٦ بوصات الى ٧٢ بوصة بزيادة قدرها ٦ بوصات ، بافتراض أن عجلة الجاذبية الأرضية هي ٨٨ . ر ٣٨٦ بوصة / ث٢ .

٣٨- يتم تحويل الدرجة X بالنظام «الستني» الى الدرجة XR بالنظام «الدائري» باستخدام القانون :

$XR = \frac{X.PI}{180}$

حيث PI هي النسبة التقريبية 3.14159

أكتب برنامج لقراءة قيم X بين ٩٠° ، ١٨٠° وكتابة قيم XR المناظر بحيث :

- تزيد كل درجة عن سابقتها بمقدار ١٠°
- يتم استخراج النتائج على الصورة التالية :

bbbbbbbbbb DEGREES b CONVERSION b TABLE

X XR

xxx xx.xxxxxx

٣٩- اقصى ارتفاع H بالأقدام يمكن أن تصل اليها قذيفة نارية تطلق بزاوية a مع المستوى الأفقي تعطي من العلاقة :

$H = \frac{V^2 \ Sin^2 \ a}{2g}$

حيث v هي السرعة الابتدائية للقذيفة محسوبة بالقدم / ث ، كما أن الزاوية a محسوبة بالتقدير الدائري وتمثل g عجلة الجاذبية بالقدم / ث٢ . فاذا كانت السرعة الابتدائية للقذيفة ثابتة وقدرها ١٠٠ قدم / ث . أكتب برنامج لحساب الزاوية a التي تصل فيها القذيفة الى أقرب ارتفاع قدره ١٠٥ قدما ، علما بأن الزاوية a يتم قرائتها بالتقدير الستينى وتتغير من ١٠ درجات الى ٩٠ درجة بزيادة قدرها ٥ درجات .

٤٠- يمكن حساب عزم القصور الذاتي m لمستطيل أبعاده b , a حول محوره من العلاقة :

$m = a^3 b^3 / 6 (a^2 + b^2)$

أكتب برنامج لقراءة قيم المتغيرات b , a وحساب القيمة m المناظر لتلك القيم .

٤١- يمكن حساب جملة مبلغ أساسه P وتم استثماره n من السنوات بعائد سنوي معدله i (معبرا عنه في صورة رقم حقيقي) ، وبحيث يحسب العائد m من المرات كل عام ، من العلاقة :

$$A = P \left(1 + \frac{i}{m}\right)^{n.m}$$

فاذا كان هناك ١٠٠٠ عميل يريدون معرفة جملة مبالغهم بعد عشر سنوات $(n = 10)$ وبمعدل عائد ٦٪ $(i = 0.06)$ ويتم حساب العائد كل $\frac{1}{4}$ سنة $(m = 4)$. أكتب برنامج لقراءة رقم كل عميل (xxxxxx) وأساس مبلغه المستثمر (xxxxxxxx.xxx) وطباعة رقم العميل وأساس مبلغه المستثمر وجملة ذلك المبلغ في نهاية العشر سنوات . رتب طباعة البيانات السابقة بكتابتها تحت عنوان مناسب .

٤٢- اللوغاريتم الطبيعي لأي عدد بين صفر ، ٤ يمكن الحصول عليه بالتقريب من المتسلسلة اللانهائية .

$$Ln\,(X) \approx 2 \quad \left[\frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \ldots.\ \frac{1}{n}\left(\frac{x-1}{x+1}\right)^n\right]$$

حيث n عدد فردي :

أكتب برنامج جزئي فرعي لحساب بعض القيم في هذا المدى (٠ - ٤) ، وأوجد عدد الحدود التي يجب إستخدامها في المتسلسلة السابقة كي يكون الخطأ المطلق بين القيمة المحسوبة بالمتسلسلة ونظيرتها المحسوبة عن طريق الدوال سابقة التخزين أقل من أو يساوي ١٠$^{-٥}$.

٤٣- يمكن حساب قيمة e^x بطريقة تقريبية وتكرارية باستخدام العلاقة :

$$e^x = 1 + \sum_{i=1}^{N} \frac{x^i}{i\,!}$$

أكتب برنامج جزئي فرعي لحساب تلك الدالة مستخدما قيما مختلفة للمتغير x . في كل حالة أوجد عدد الحدود التي يجب جمعها كي يكون الخطأ المطلق بين القيمة المحسوبة بالصيغة السابقة ونظيرتها المحسوبة عن طريق الدوال سابقة التخزين أقل من أو يساوي ١٠$^{-٥}$.

٤٤- صيغة استيرلنج التقريبية لحساب مضروب عدد ما N ، (N !) يعطى من العلاقة :

$$N\,! \approx 2\,\pi.n'(n^n\,e^{-n})$$

أكتب برنامج جزئي لدالة لحساب تلك الصيغة لاعداد صحيحة موجبة بين ٢ ، ٩ .

٤٥- في التمرين رقم ١ ، أكتب كل جزء في صورة برنامج جزئي لدالة .

٤٦- في التمرين رقم ٢٦ ، أكتب برنامج لقراءة المصفوفة الخطية X ، ثم أكتب برنامج جزئي فرعي لاستخراج أكبر الأعداد وكذلك أصغرها ومواقعهما في تلك المصفوفة .

٤٧- في التمرين رقم ٢٨ ، أكتب برنامج لقراءة المصفوفة A_{MXN} . أكتب برنامج جزئي فرعي لاستخراج أكبر وأصغر عنصرين في تلك المصفوفة وموقعهما في المصفوفة .

٤٨- في التمرين رقم ٣٩ ، أعد كتابة البرنامج بحيث يتم حساب قيمة H عن طريق برنامج جزئي لدالة .

٤٩- كتب برنامج جزئي لدالة لحساب العلاقة التالية :

$$1)\ S = A \cos x - B \cos y + c - \cos(x-y)$$

حيث :

$$A = \tan^{-1} y ,\ B = \sin^{-1}\left(\frac{x}{y}\right),\quad C = A.B/(A+B)$$

$$2)\ V = \frac{1}{4}\pi d^2 \sqrt{\left(\frac{gd}{4ft}\right)\frac{1}{wa_1}\left(a_1^2 - a^2\right)}$$

٥٠- المصفوفة الخطية LIST تحتوي على ٥٠٠ اسما مختلفا وكل اسم مكون من ٢٠ حرفاً ورقما . أكتب برنامج جزئي فرعي يُعطى له اسم ما يتم قراءته عن طريق البرنامج الرئيسي كي يقوم البرنامج الجزئي الفرعي بمقارنة هذا الاسم بالاسماء الموجودة داخل المصفوفة الخطية . وتكون نتيجة البحث في صورة متغير يأخذ القيمة YES في حالة العثور على الاسم داخل المصفوفة ، والقيمة NO في حالة عدم العثور عليه .



قاموس المصطلحات

قاموس المصطلحات

«ب»

«ت»

«ح»

«خ»

«د»

«ر»

«س»

«ش»

«ص»

«ط»

«ع»

«غ»

«ف»

«ق»

«ك»

«ل»

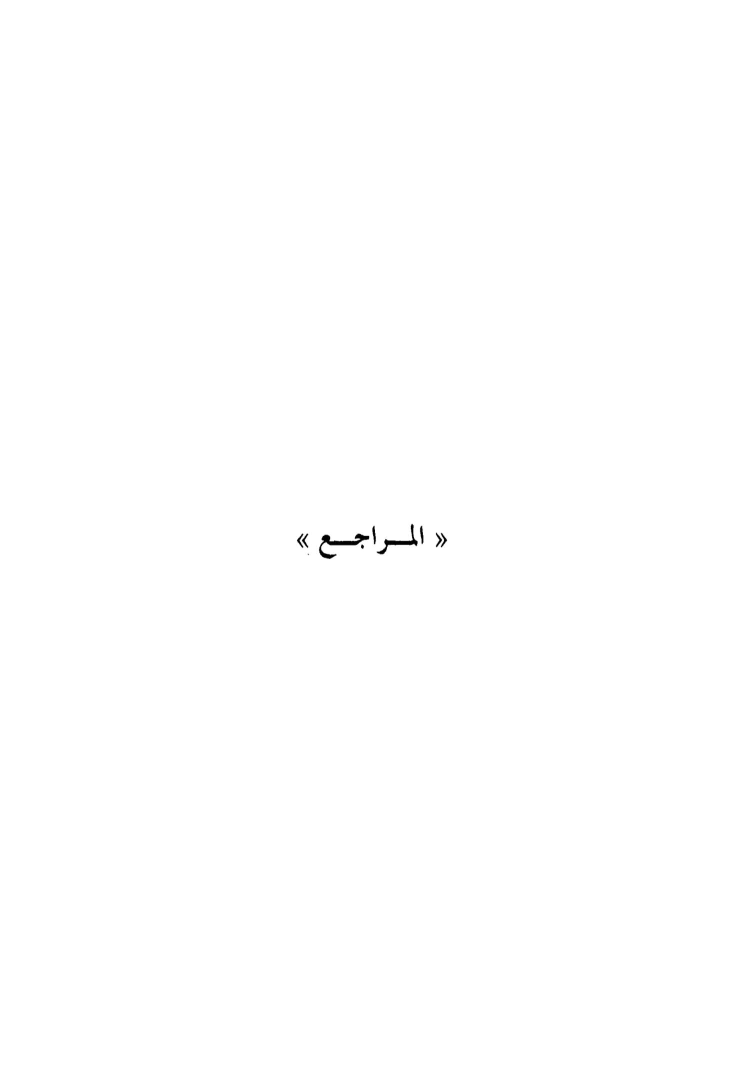
«م»

«ن»

«هـ»

«و»

« ي »



« المـــراجـــع »

1 – Frank L. Friedman and Elliot B. Koffman : Problem Solving and Structured Programming in Fortran, 2nd ed., Addison- Wesley, 1981.

2 – Hossam El-Beblawi : Computer Glossary, Studies in Automatic Al-Maaref Estab., 1975.

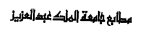
3 – Meissner / Organick : Fortran 77 Featuring Structured Programming, Addison-Wesely, 1980.

4 – Rich Didday and Rex Page : Fortran for Humans, 3rd ed., West, 1981.

5 – Robert H. Hammond; William B. Rogers and Byard Houck : Introduction to Fortran IV, 2nd ed., McGraw Hill,1978.

6 – William F. Schallert and Carol Reedy Clark : Programming in Fortran, Addison Wesley, 1979.

7 – William S. Davis : Fortran Getting Standard, Addison- Wesley, 1981.


مطابع جامعة الملك عبدالعزيز



مطابع جامعة الملك عبد العزيز